EFFICIENT SOLVERS FOR IMAGE DEBLURRING PROBLEM AND STOCHASTIC DARCY'S EQUATIONS

BY

ADEL MOHAMMED YAHYA AL-MAHDI

A Dissertation Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

MATHEMATICS

DECEMBER, 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by **ADEL MOHAMMED YAHYA AL-MAHDI** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN MATHEMATICS**

Dissertation Committee

[illegible]
Dr. Faisal Fairag (Advisor)

[illegible]
Prof. Fiazud Din Zaman (Co-Advisor)

[illegible]
Prof. Mohamed El-Gebeily (Member)

[illegible]
Prof. Kassem Mustapha (Member)

[illegible]
Dr. Muhammad Yousuf (Member)

[illegible]
Dr. Husain Salem Al-Attas
Department Chairman

[illegible]
Prof. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
ATE STUDIES

Date: 28/12/15

I dedicate my Dissertation work to my family. A special feeling of gratitude to my loving parents, my wife, my son, my daughters, my brothers, my sisters.

ACKNOWLEDGMENTS

I am most grateful to Almighty ALLAH, the Beneficent, the Merciful, for enabling me to complete this work. Peace and blessings of ALLAH be upon his Last messenger Mohammed (Sallallah-Alaihe-Wasallam) and his family, who guided us to the right path.

First and the foremost acknowledgments are due to the King Fahd University of Petroleum and Minerals and to the Department of Mathematical Sciences for supporting my research work.

I wish to express My deep appreciation and heartfelt gratitude to Dr. Faisal Fairag, my thesis advisor for his unfailing encouragement, advice and suggestions through this work and Co. advisor Prof. F. D. Zaman who guided me with his dedicated attention, expertise and knowledge throughout this research. I also wish to thank my Committee Members, Prof. Mohamed El-Gebeily, Prof. Kassem Mustapha and Dr. Muhammad Yousuf for their constructive guidance and support.

And last, but not the least, my cordial thanks and appreciation are due to my parents, my wife and children, brothers, friends and all members of my family who always support me with their love, patience, encouragement and constant prayers.

TABLE OF CONTENTS

List of Tables

List of Figures

DISSERTATION ABSTRACT

Name:	**Adel Mohammed Yahya Al-Mahdi**
Title:	**Efficient Solvers For Image Dublurring Problem And Stochastic Darcy's Equations**
Major Field:	**Mathematics**
Date of Degree:	**December, 2015**

We consider the numerical solutions of two large and ill-conditioned linear systems which arise in applications. The first system arises when the total variational regularization is applied to solve an ill-posed problem (image deblurring problem) while the second system results from the discretization of the $([L^2(D)]^2 \times L^2_{\mathbb{P}}(\Omega)) \otimes (H^1(D) \cap L^2_0(D) \otimes L^2_{\mathbb{P}}(\Omega))$ *formulation for the stochastic Darcy's equations. In each system the coefficient matrix has huge size and large condition number. These properties of the coefficient matrices make any iterative method for such a system very slow. To overcome this problem, we introduce several new preconditioners for such a system to accelerate the convergence of the iterative method that we will use. These preconditioners are of Murphy, Golub and Wathen type. We show that the preconditioned matrices have eigenvalues clustering behavior. This behavior leads to large reduction in the number of iterations. We test the performance of the preconditioned iterative methods through several numerical examples.*

ملخص بحث
درجة الدكتوراة في الفلسفة

الاســـــــــم: **عادل محمد يحي المهدي**

عنوان الرسالة: حلول فعالة لمشاكل توضيح الصور الرقمية ومعادلات **دارسي** بمعامعلات عشوائية**.**

التــخـصـــص : الرياضيـــات.

تاريخ التخرج : ديسمبر 2015

في هذه الرسالة اهتمينا بالحلول العددية لنوعين من انواع المعادلات. الاولى معادلة تفاضلية-تكاملية غير خطية ناتجة عن مشاكل توضيح الصور الرقمية والمعادلة الثانية هي معادلة تفاضلية (معادلات دارسي) بمعاملات عشوائية. هذه الانواع من المعادلات تتطلب حل نظام خطي كبير جدا. مصفوفة معامل هذا النظام لها رقم شرطي (condition number) كبير جدا مما يجعل اي طريقة تكرارية لهذا النظام بطيئة جدا. ولمعالجة هذا البطئ اقترحنا استخدام مهيئات لهذه النظم الخطية لكي تقوم بتسريع تكرار الطرق . هذه المهيئات من نوع مهيئات مشابهة ل **ميرفي-غلوب-وثن** . القيم الذاتية للمصفوفات المهيئة لديها صفة التجميع وهذا السلوك التجميعي يؤدي الى خفض كبير جدا في عدد التكرارات. في هذا البحث اختبرنا اداء المهيئات المقترحة من خلال عدة امثلة عددية.

Chapter 1

INTRODUCTION

1.1 Motivation

In many applications, most of discretized problems lead to linear system of equations of the form

$$\underbrace{\begin{bmatrix} D & {B_1}^T \\ B_2 & -C \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} u \\ p \end{bmatrix}}_{\text{x}} = \underbrace{\begin{bmatrix} f \\ g \end{bmatrix}}_{\text{b}}. \tag{1.1.1}$$

The above system is called generalized saddle point system (saddle point system if $C = 0$). This system appears in many fields such as the following situations:

- solving partial differential equations (pdes) by mixed finite elements methods.
- image reconstruction problems.
- discretizing Darcy and non-Darcy equations.
- optimization problems.
- discretizing Stokes and Navier-Stokes problem.
- finance, economics and optimal control.

For the other areas where saddle point problems naturally arise, we refer to [11], [100]. The system (1.1.1) is often indefinite and ill-conditioned. These properties come from the discretization methods like finite element, finite volume and finite difference methods. Due to these properties, the numerical solution of such systems represent a big challenge for those interested in solving these systems and thus this is an active area research.

In this dissertation, we consider two important saddle point systems which arise in applications. The first system arises when total variational (TV) regularization is applied to solve an image deblurring problem while the second one results from the discretization of the $[L^2(D)]^2 \otimes L^2_{\mathbb{P}}(\Omega)$ and $(H^1(D) \cap L^2_0(D)) \otimes L^2_{\mathbb{P}}(\Omega)$ formulation for stochastic Darcy's equations.

The coefficient matrices of both systems have huge size and large condition numbers. The reason for the huge size of the first system (image deblurring problem) is that the lower resolution image of 256×256 pixel array has a corresponding matrix of 256^4 entries.
The reason for the huge size of the second system (stochastic Darcy's equations) is that the coefficient matrix is a kronecker product of two block matrices of which one comes from the deterministic part while the second results from the stochastic part.

There are two classes of linear solvers. The first class is based on the direct methods while the second on the iterative methods. It is known that direct methods like LU and Cholesky factorization can be used if the solvable system is of a reasonable size. This is because solving linear systems using direct methods requires $O(n^3)$ arithmetic operations, where n is the length of the solution vector x. Hence, for the two linear system which studied here, direct methods are not applicable.
In this case, we use suitable iterative methods that are usually based on Krylov subspace methods. But the problem is that the convergence of these methods is slow in the case of ill-conditioning matrices.
To overcome the slowness of the convergence, we find suitable preconditioning matrices, so that the preconditioned matrices have good spectral properties.

In the following two sections, we present the two saddle point systems which we aim to study in this dissertation.

1.2 Image deblurring problem

The first saddle point system that we will study in this dissertation is of the form

$$\begin{bmatrix} \alpha D & -\alpha B \\ -\alpha B^T & -K^*K \end{bmatrix} \begin{bmatrix} V \\ U \end{bmatrix} = \begin{bmatrix} 0 \\ -K^*Z \end{bmatrix}. \tag{1.2.1}$$

The above system arises when total variational regularization is applied to solve an ill–posed problem (image deblurring problem).

The importance of such system is due to the wide applications of the image deblurring. For instance Saher needs to remove the blur from the car image taken while the camera is shaking or in radar imaging and tomography one needs to remove the effect of imaging systems response. Other applications arise in medical images where deblurring is an essential requirement.

The system (1.2.1) is in the generalized saddle point form. Its coefficient matrix is of huge size and it is highly ill-conditioned. The (2,2)–block of this matrix has the block Toeplitz with Toeplitz block (BTTB) structure. Moreover, in this system, the negative Shur complement of its coefficient matrix is the sum of two matrices. The first matrix, K^*K, is dense and comes from the discretization of a compact integral operator while the second (sparse) matrix, $L = (B^T D^{-1} B)$, is called the regularization matrix results from the discretization of a diffusion operator.

1.2.1 Overview of image deblurring

When the coefficient matrix of the system (1.2.1) is symmetric, indefinite, large and ill-conditioned, MINRES is the suitable iterative method. However, a preconditioner

is needed to achieve the fast convergence. MINRES with such a preconditioner is called PMINRES. However, not any preconditioner can be used.
What is needed is an efficient preconditioner. The efficiency can be tested through the number of iterations, the CPU-time and the clustering behavior of the eigenvalues of the preconditioned matrix.
Our starting point here is that the Schur complement of the matrix of the system (1.2.1) contains a product of a Toepelitz matrix with Toepelitz blocks (BTTB) and its transpose. This product may not be a BTTB.

We approximate this product by several approaches. In the first one, we approximate it by a symmetric BTTB matrix. In the second approach, we use the Strang circulant approximation of a BTTB matrix. The last approach uses the best circulant approximation for the BTTB matrix. Both of theses approximations alow us to use the Fast Fourier Transform (FFT) for matrix-vector multiplication. This multiplication is needed in PMINRES computation because in each PMINRES iteration we need to solve a linear system of the form $Px = y$ where P is the preconditioner matrix. So FFT reduces the cost of the computation from $O(n^2)$ arithmetic operations to $O(n \log n)$ arithmetic operations.

As a consequence of these three approximation, we develop three efficient block diagonal preconditioners. These preconditioners are of Murphy, Golub and Wathen (MGW) type and they depend on these three approximations of the product of the BTTB matrix and its transpose. We investigate the efficiency of these preconditioners by several numerical computations in terms of CPU-time, iteration numbers and the quality of the reconstructed images. In the following section, we present the second saddle point system considered in this dissertation.

1.3 Stochastic Darcy's equations

The second saddle point system which we study in this dissertation is in the form

$$\begin{bmatrix} \hat{A} & \hat{B}^T \\ \hat{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f} \end{bmatrix}. \tag{1.3.1}$$

The above system occurs in the discretization of the $[L^2(D)]^2 \otimes L^2_{\mathbb{P}}(\Omega)$ and $(H^1(D) \cap L^2_0(D)) \otimes L^2_{\mathbb{P}}(\Omega)$ formulation for Darcy's equation with stochastic coefficients. The result of this discretization is a huge and ill-conditioned linear system (1.3.1). The reason for the huge size of this system is that the coefficient matrix and the right hand side are a kronecker product of two block matrices, one from the deterministic part while the other from the stochastic part. This kind of equations is important in petroleum industry and in describing the flow of fluid in pours media. For these reasons, we study the numerical solution of this problem.

1.3.1 Overview of the stochastic Darcy's equations

As we mentioned above, the linear system (1.3.1) arises from discretaizing the mixed formulation of Darcy's equations with random data. In this discretaization, we use a truncated Karhunen-Loève (K-L)-expansion to represent this random coefficient. We use the stochastic Galerkin finite element method (SGFEM). In this method, the deterministic part is discretized using classical mixed finite element methods and the stochastic part by using a tensor product (TP) polynomial space. The highly structured linear system that results from this discretization means that Krylov subspace methods with a suitable preconditioner as linear solver is extremely effective here. Since the coefficient matrices of the systems (1.3.1) and (1.2.1) are symmetric and indefinite, a suitable iterative method is MINRES.

Hence, we propose and analyze several block-diagonal preconditioners. These preconditioners are also of Murphy, Golub and Wathen type and are based on the $[L^2(D)]^2 \otimes L^2_{\mathbb{P}}(\Omega)$ and $(H^1(D) \cap L^2_0(D)) \otimes L^2_{\mathbb{P}}(\Omega)$ spaces of the Darcy's velocities and the pressure respectively.

The attractive properties of this choice of the discrete spaces is that the (1,1) block in the coefficient matrix has the diagonal structure and the Schur complement of the coefficient matrix of this system is the well known discrete Laplacian matrix. These nice properties lead to a reduction in the cost of the computation and give a solution with less number iterations .

1.4 Organization of the dissertation

This dissertation is organized as follows: In Chapter 2, we present some fundamental definitions and notations related to the image reconstruction problem and stochastic Darcy's equations. In Chapter 3, we present the mathematical model behind the image deblurring problems. We derive several preconditioners and implement them in Chapter 4. In Chapter 5, we present, analyze and implement the stochastic Galrkin finite element method for the stochastic Darcy's equations. In Chapter 6, we propose several preconditioners for both deterministic and stochastic Darcy equations. We give conclusions of this study and propose some future directions in Chapter 7. Finally, the Matlab codes which was used in our computations is given in Chapter 8.

Chapter 2

PRELIMINARIES

2.1 Introduction

In this chapter, we present some definitions, spaces and other concepts that we need in the next chapters of this dissertation. Since we are interested in saddle point problems, we start by giving some properties, factorizations, inverse and solvability of the saddle point systems in the following sections and then we introduce some preliminaries and notations related to the image deblurring problem and finally to the stochastic problem.

2.2 Saddle point matrices and their properties

In this section, we give some properties of the matrix A given in (1.1.1), which is in the saddle point form.

2.2.1 Factoring saddle point matrices and their Schur complements

In this subsection, we aim to give some factorizations for the generalized saddle point matrix A in the non-symmetric case , (we assume that D is invertible (non-singular)), as follows:

$$A = \begin{bmatrix} D & B_1{}^T \\ B_2 & -C \end{bmatrix} = \begin{bmatrix} I & 0 \\ B_2 D^{-1} & I \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & -S \end{bmatrix} \begin{bmatrix} I & D^{-1}B_1{}^T \\ 0 & I \end{bmatrix}. \tag{2.2.1}$$

Here, $S = (C + B_2 D^{-1} {B_1}^T)$ is called the positive *Schur complement* of the block D in the big-matrix A. Also A has the following factorizations

$$A = \begin{bmatrix} D & {B_1}^T \\ B_2 & -C \end{bmatrix} = \begin{bmatrix} D & 0 \\ B_2 & -S \end{bmatrix} \begin{bmatrix} I & D^{-1}{B_1}^T \\ 0 & I \end{bmatrix} \tag{2.2.2}$$

and

$$A = \begin{bmatrix} D & {B_1}^T \\ B_2 & -C \end{bmatrix} = \begin{bmatrix} I & 0 \\ B_2 D^{-1} & I \end{bmatrix} \begin{bmatrix} D & {B_1}^T \\ 0 & -S \end{bmatrix} \tag{2.2.3}$$

2.2.2 Solvability conditions

We see that from (2.2.1)-(2.2.3), D is needed to be nonsingular. Moreover, A is invertible if S is also invertible. However, the non singularity of $S = (C + B_2 D^{-1} {B_1}^T)$ is .... by putting some conditions on the component matrices B_1, B_2, D and C.

In (2.2.1), if $C = 0$, D is symmetric positive definite (spd) and $B_1 = B_2$, then we get the so called standard symmetric saddle point matrix in which the Schur complement is $S = BD^{-1}B^T$. It is clear that the Schur complement S, and thus the saddle point matrix A, is invertible if B has full row rank. Now, if $C \neq 0$ is symmetric positive semidefinite (sps), D is spd and $B_1 = B_2 = B$, then, again $S = (C + BD^{-1}B^T)$ is sps. Moreover, it is positive definite and hence invertible if $ker(C) \cap ker(B^T) = \{0\}$. It is obvious that sufficient conditions for the invertibility are either C be positive definite **or** B has full row rank. The above discussion can be summarized in the following theorem.

Theorem 1 *Let D be an spd matrix, C be sps and $B_1 = B_2 = B$. If $ker(C) \cap ker(B^T) = 0$, then the matrix A is invertible. In particular, if B has full row rank,* *A is invertible.*

If D is indefinite, as in the following example

$$\left[\begin{array}{cc|c} 1 & 0 & -1 \\ 0 & -1 & 1 \\ \hline -1 & 1 & 0 \end{array}\right] = \begin{bmatrix} D & B^T \\ B & 0 \end{bmatrix}, \tag{2.2.4}$$

then A may be singular, even if B has full rank. However, A will be nonsingular if D is positive definite on $ker(B)$. In the case of D is symmetric positive semidefinite, we have the following theorem.

Theorem 2 *Let $C = 0$, D be sps and $B_1 = B_2 = B$ has full rank. Then a necessary and sufficient condition for the invertibility of the saddle point matrix A is $ker(D) \cap ker(B) = \{0\}$.*

Full discussion about the solvability conditions can be found in [11].

2.2.3 Inverse of a saddle point matrix

Assume that D is invertible, then the saddle point matrix A is invertible if the Shur compliment matrix $S = (C+B_2D^{-1}B^T)$ is invertible, moreover, we have the following:

$$A^{-1} = \begin{bmatrix} D & {B_1}^T \\ B_2 & -C \end{bmatrix}^{-1} = \begin{bmatrix} D^{-1} - D^{-1}{B_1}^T S^{-1} B_2 D^{-1} & D^{-1}{B_1}^T S^{-1} \\ S^{-1}B_2D^{-1} & -S^{-1} \end{bmatrix} \tag{2.2.5}$$

For other cases (for example when D is singular but C is nonsingular) see [11].

2.2.4 Eigenvalues of the saddle point matrix

Assume that the matrix D is spd, C is sps (it could be zero) and $B_1 = B_2 = B$ has full row rank. Then

$$\begin{bmatrix} I & 0 \\ -BD^{-1} & I \end{bmatrix} \begin{bmatrix} D & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} I & -D^{-1}B^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} D & 0 \\ 0 & -S \end{bmatrix} \tag{2.2.6}$$

where the Schur compliment matrix $S = (C + BD^{-1}B^T)$ is spd. Then the saddle point matrix A is congruent to the block diagonal matrix on the right hand side of the above equation. The above congruence is called Sylvesters Law of Inertia. From this congruence, it follows that the number of the positive and negative eigenvalues of A are the same as that of the block diagonal matrix given in the right hand side.

2.3 Krylov subspace iterative methods

Often Krylov subspace iterative methods are used to compute iterates solutions x_k of the linear system $Ax = b$ for which

$$x_k - x_0 \in \mathbb{K}_k(A, r_0), \ k = 1, 2, ..., \tag{2.3.1}$$

where

$$\mathbb{K}_k(A, r_0) = span\{r_0, Ar_0, A^2r_0, ..., A^{K-1}r_0\}, \ k = 1, 2, ...,$$

is called the Krylov subspace associated with A and r_0. In (2.3.1), x_0 is the initial guess (some times it is taken to be zero). Thus Krylov subspace iterative methods require just one matrix-vector product computation at each iteration. $r_0 = b - Ax_0$ is called the residual vector associated with x_0; in general $r_j = b - Ax_j; \ j = 0, 1, ...$

If $x_0 = 0$, then

$$x_k \in \mathbb{K}_k(A, b),\ \ k = 1, 2, ...$$

Thus the iterates solutions and residuals of every Krylov subspace method satisfy

$$x_k - x_0 = \sum_{j=0}^{k-1} \alpha_j A^j r_0,$$

for some coefficients α_j. Hence

$$x_k = x_0 + q(A)r_0, \tag{2.3.2}$$

where q is the polynomial of degree $k-1$ with $q(z) = \sum_{j=0}^{k-1} \alpha_j z^j$. Multiplying (2.3.2) by A and then subtracting from b, we obtain

$$b - Ax_k = b - Ax_0 - Aq(A)r_0,$$

and thus the residuals

$$r_k = r_0 - Aq(A)r_0 = P(A)r_0, \tag{2.3.3}$$

where

$$P(z) = 1 - z\sum_{j=0}^{k-1} \alpha_j z^j = 1 - \sum_{j=1}^{k} \alpha_{j-1} z^j,$$

is a polynomial of degree k which satisfies $P(0) = 1$. All Krylov subspace methods are thus described by (2.3.3).

Different Krylov subspace methods can be characterized by the properties of the matrix A as follows:

When A is spd (so that $\| v \|_A = (v^t A v)^{\frac{1}{2}}$ defines a vector norm or energy norm), the conjugate gradient method (CG) [54] requires only the one matrix-vector multiplication by A and it minimizes the A-nor of the error $\| x - x_k \|_A$ over the Krylov subspace.

When A is symmetric but indefinite, $(v^t A v)$ takes both positive and negative values, so a norm cannot be defined as for the conjugate gradient method. The Krylov subspace method of choice for symmetric indefinite systems is the minimum residual (MINRES) method [77]. It takes one matrix-vector product with A and it minimizes the Euclidean norm of the residual, $\| r_k \|_I = ({r_k}^t r_k)^{\frac{1}{2}}$.

When A is non-symmetric, there is not such an obvious method of choice, hence several Krylov subspace methods are widely used. The most popular is GMRES [86] which, similarly to MINRES, computes iterates that minimize the Euclidean norm of the residual, but by contrast to MINRES requires an increasing amount of computation and storage at each successive iteration to achieve this. Thus GMRES can be a good method if only a few iterations are needed to achieve acceptable convergence, this might be the case if one has a good preconditioner, but it is not practical if many iterations are required.

Anyway, an appropriate iterative method will compute a sequence of vectors $x_1, x_2, ...$ which converge rapidly from any starting guess, x_0, to the solution x of the system $Ax = b$. At each iteration, only a matrix-vector product with A needs to be computed. Unfortunately, Krylov subspace methods are very slow with an ill-conditioned linear system of equations. One technique to overcome this slowness is using an appropriate preconditioner. Preconditioners are overwhelmingly used with Krylov subspace iterative methods (see for examples [97], [49], [85], [69], [52], [32], [62], [76]).

2.4 Preconditioning technique

For the successful use of iterative methods, we have to use a preconditioning technique. In 1948, Turing was the first one who used the term of preconditioning in his paper [95]. In [36], Evans used the term of preconditioning in connection with iterative methods. In [17] Cesari was the first one who used preconditioning for reducing the condition number in order to improve convergence of some iterative methods.

The term preconditioning refers to transforming the system (1.1.1) into another system that has a smaller condition number. Consider the matrix P to be the preconditioner matrix for the matrix A given in (1.1.1), then the linear system

$$P^{-1}Ax = P^{-1}b, \tag{2.4.1}$$

has the same solution as (1.1.1) but (5.4.9) may be faster than (1.1.1). Moreover, the preconditioning makes the computing time for solving (5.4.9) less than for solving (1.1.1). A good preconditioner which accelerates the convergence needs to be easy to construct and cheap to invert. Moreover, the preconditioned matrix should have eigenvalues clustering behavior. Many preconditioners in [11] are developed for a special linear system such as a saddle point problem. For the improvement of the preconditioning techniques for general linear systems, we refer to [10] and [100]. For the types of the preconditioners, there are mainly two classes: block preconditioners and constraint preconditioners (see [10] and [11]). Block diagonal preconditioners have been studied by Murphy, Golub and Wathen in [71] and later by Ipsen in [55] and by de Sturler, E. and Liesen in [25]. There are many studies for block diagonal preconditioners introduced by Silvester and Wathen [89]. Analysis of these preconditioners have been given in [78], [59], [61] and [38].

As mentioned in the above discussion, the numerical solutions of such saddle point systems represent a big challenge and they have made the research in this area is very active.

Since we use PMINRES as a linear solver for both systems, we give a short review of PMINRES method.

2.5 PMINRES method

Suppose we need to solve the linear system $Ax = b$ where A is a symmetric and indefinite saddle point matrix and suppose that a spd- preconditioner is considered

$$P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}, \tag{2.5.1}$$

It is known that PMINRES generates a sequence of iterates solutions x_k which belong to the following Krylov space

$$K_k = span\{P^{-1}r_0, (P^{-1}A)P^{-1}r_0, ..., (P^{-1}A)^{k-1}P^{-1}r_0\}, \tag{2.5.2}$$

with minimization the norm of the $k-th$ residual

$$\| r_k \|_{P^{-1}} = \| b - Ax_k \|_{P^{-1}} = \min_{x \in K_k} \| b - Ax \|_{P^{-1}}, \tag{2.5.3}$$

and $\| v \|_{P^{-1}} = v^T P^{-1} v$. The PMINRES convergence estimate [31] is given by

$$\frac{\| r^{(k)} \|_{P^{-1}}}{\| r^{(0)} \|_{P^{-1}}} \leq \min_{q_k \in \Pi_k,\ q_k(0)=1} \max_{\lambda \in \sigma(P^{-1}A)} | q_k(\lambda) | \tag{2.5.4}$$

where Π_k is the space of all polynomial of degree less than or equals k and $\sigma(P^{-1}A)$ is the spectrum of the preconditioned matrix $(P^{-1}A)$. To minimize the right hand side of the above inequality (2.5.4), it is desirable to cluster both the positive and negative eigenvalues of the preconditioned matrix $P^{-1}A$. This clustering guarantees convergence with few iterations. In the following sections, we give some definitions and notations related to the image deblurring problems.

2.6 Fourier transform (FT) and convolution integral

In this section, we present some definitions of the continuous and discrete one and two-dimensional Fourier Transforms.

2.6.1 (1-D) Fourier transform and convolution theorem

Given any function f defined on $\mathbb{R}$ (possibly complex-valued), the , one-dimensional, continuous, FT is defined by

$$(\mathcal{F}u)(\omega) = \int_{\mathbb{R}} u(x) e^{-2\pi \hat{\imath} x \omega} dx, \quad \omega \in \mathbb{R}, \tag{2.6.1}$$

with $\hat{i} = \sqrt{-1}$ and its inverse is given by

$$(\mathcal{F})^{-1}(v)(x) = \int_{\mathbb{R}} v(\omega)e^{2\pi\hat{i}x\omega}d\omega, \quad x \in \mathbb{R}. \tag{2.6.2}$$

The convolution of two functions u, v is defined by

$$(u * v)(x) = \int_{\mathbb{R}} u(x-y)v(y)dy, \quad x \in \mathbb{R}. \tag{2.6.3}$$

The FT and its inverse of the convolution is given by

$$\mathcal{F}(u * v) = \mathcal{F}(u) \cdot \mathcal{F}(v), \tag{2.6.4}$$

$$\mathcal{F}^{-1}(u * v) = \mathcal{F}^{-1}(u) \cdot \mathcal{F}^{-1}(v), \tag{2.6.5}$$

where $\cdot$ denotes point-wise multiplication. Next, we give definitions for a discrete Fourier transform (DFT) as follows:

Definition 1 *The discrete Fourier transform of a sequence* $\{u_l\}_{l=0}^{n-1}$ *is defined by*

$$[\mathcal{F}(u)]_k = \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} u_l e^{\frac{-2\hat{i}\pi kl}{n}}, \ k = 0, ..., n-1, \tag{2.6.6}$$

Note that, the DFT can be expressed as a matrix-vector product, $\mathcal{F}\{u\} = Fu$, where $F \in \mathbb{C}^{n \times n}$ is the Fourier matrix. It has the components

$$[F(u)]_{kl} = \frac{e^{\frac{-2\hat{i}\pi kl}{n}}}{\sqrt{n}}, \quad 0 \leq k, l \leq n-1. \tag{2.6.7}$$

The inverse DFT is given by

$$[\mathcal{F}^{-1}(v)]_i = \frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} v_l e^{\frac{2\hat{\imath}\pi kl}{n}} = [F^* v], \ k = 0, ..., n-1, \tag{2.6.8}$$

where $*$ denotes the conjugate transpose of a matrix.

2.6.2 (2-D) Fourier transform and convolution theorem

Definition 2 *The two-dimensional continuous FT of a function u defined on $\mathbb{R}^2$ (it could be a complex-valued function) is*

$$(\mathcal{F}u)(\omega) = \int_{\mathbb{R}^2} u(x) e^{-2\hat{\imath}\pi x^T \omega} dx, \quad \omega \in \mathbb{R}^2, \tag{2.6.9}$$

and its inverse is given by

$$(\mathcal{F}^{-1}v)(x) = \int_{\mathbb{R}^2} v(\omega) e^{2\hat{\imath}\pi x^T \omega} d\omega, \quad x \in \mathbb{R}^2, \tag{2.6.10}$$

Definition 3 *The two-dimensional convolution integral is defined by*

$$(u * v)(x) = \int_{\mathbb{R}^2} u(x-y) v(y) dy, \quad x \in \mathbb{R}^2. \tag{2.6.11}$$

Definition 4 *The two-dimensional DFT is the matrix given by*

$$[\mathcal{F}(u)]_{kl} = \frac{1}{\sqrt{n_x n_y}} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} u_{i,j} e^{-2\hat{\imath}\pi(ki/n_x + lj/n_y)}, \tag{2.6.12}$$

where $0 \le k \le n_x - 1, \ 0 \le l \le ny - 1$ and its inverse can be obtained by replacing $-\hat{\imath}$ by $\hat{\imath}$ in (2.6.12).

2.7 Fast Fourier transform

In this section, we present definitions of the one-dimensional and two-dimensional fast Fourier transform (FFT).

2.7.1 (1-D) Fast Fourier transform

To implement (2.6.6), we use the conventional matrix-vector multiplication. In this case, it will cost $O(n^2)$ operations, where n is the length of the vector that we need to transform. The FFT algorithm, which was developed by Cooley and Tukey [26], reduces this computational cost to $O(n \log n)$.

Definition 5 *given any $\boldsymbol{u} = (u_0, u_1, ..., u_{n-1}) \in \mathbb{C}^n$, the Fast Fourier transform (**fft**) is defined by*

$$[\boldsymbol{fft}(\boldsymbol{u})]_i = \sqrt{n}[\mathcal{F}(u)]_k = \sum_{l=0}^{n-1} u_l e^{\frac{-2\hat{i}\pi kl}{n}}, \quad k = 0, ..., n-1. \tag{2.7.1}$$

*The inverse of (**fft**) is given by*

$$[\boldsymbol{ifft}(\boldsymbol{u})]_i = \frac{1}{\sqrt{n}}[\mathcal{F}^{-1}(u)]_i = \frac{1}{n}\sum_{l=0}^{n-1} u_l e^{\frac{-2\hat{i}\pi kl}{n}}, \quad k = 0, ..., n-1. \tag{2.7.2}$$

The FFT and its inverse satisfy

$$\boldsymbol{fft}(u * v) = \boldsymbol{fft}(u) \cdot \boldsymbol{fft}(v), \tag{2.7.3}$$

and

$$\boldsymbol{ifft}^{-1}(u * v) = \boldsymbol{ifft}^{-1}(u) \cdot \boldsymbol{ifft}^{-1}(v), \tag{2.7.4}$$

2.7.2 (2-D) Fast Fourier transform

Two-dimensional fast Fourier transform **fft2** can be defined in analogous manner to (2.7.1)-(2.7.2) as follows

Definition 6

$$[\boldsymbol{fft2}(\boldsymbol{u})]_{kl} = \sqrt{n_x n_y}[\mathcal{F}(f)]_{kl} = \sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-1} u_{i,j} e^{-2\hat{i}\pi(ki/n_x + lj/n_y)}, \; k = 0, ..., n-1. \quad (2.7.5)$$

The inverse of (***fft2***) *is given by*

$$[\boldsymbol{ifft2}(\boldsymbol{u})]_{kl} = \frac{1}{\sqrt{n_x n_y}}[\mathcal{F}^{-1}(u)]_{kl} = \frac{1}{n_x n_y}[\boldsymbol{fft2}(\boldsymbol{u})]_{kl} = \frac{1}{n_x n_y}\sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-1} u_{i,j} e^{-2\hat{i}\pi(ki/n_x + lj/n_y)}, \quad (2.7.6)$$

for $k = 0, ..., n-1$.

2.8 Toeplitz and circulant matrices

In this section, we present definitions of the Toeplitz and circulant matrices and also of the block Toeplitz and block circulant matrices.

Definition 7 *An* $n \times n$ *matrix* T *is called Toeplitz if the entries along each diagonal are the same and has the following form*

$$T = \begin{bmatrix} \lambda_0 & \lambda_{-1} & \cdots & \lambda_{1-n} \\ \lambda_1 & \lambda_0 & \lambda_{-1} & \cdots \\ \vdots & \ddots & \ddots & \lambda_{-1} \\ \lambda_{n-1} & \cdots & \lambda_1 & \lambda_0 \end{bmatrix} \quad (2.8.1)$$

Definition 8 *A circulant matrix is a Toeplitz matrix in which each column/row is* *a circular shift of the elements in the preceding column/row. In this case, an $n \times n$ circulant matrix C has the form*

$$C = \begin{bmatrix} \delta_0 & \delta_{n-1} & \cdots & \delta_1 \\ \delta_1 & \delta_0 & \delta_{n-1} & \cdots \\ \vdots & \ddots & \ddots & \delta_{n-1} \\ \delta_{n-1} & \cdots & \delta_1 & \delta_0 \end{bmatrix} \tag{2.8.2}$$

For more information of circulant matrices and their properties, we refer to see [29].

2.8.1 Block Toeplitz and block circulant matrices

We define We give definitions of the block Toeplitz with Toeplitz block (BTTB) and block circulant with circulant block (BCCB) matrices as follows

Definition 9 *An $n_x n_y \times n_x n_y$ matrix $\boldsymbol{T}$ is called BTTB if it has the block form*

$$\boldsymbol{T} = \begin{bmatrix} \Lambda_0 & \Lambda_{-1} & \cdots & \Lambda_{1-n} \\ \Lambda_1 & \Lambda_0 & \Lambda_{-1} & \cdots \\ \vdots & \ddots & \ddots & \Lambda_{-1} \\ \Lambda_{n-1} & \cdots & \Lambda_1 & \Lambda_0 \end{bmatrix} \tag{2.8.3}$$

where each block Λ_j is an $n_x \times n_x$ Toeplitz matrix.

Definition 10 *An $n_x n_y \times n_x n_y$ matrix $\boldsymbol{C}$ is BCCB if $\boldsymbol{C}$ is BTTB first and then if each $nx \times nx$ block column/row is a circular shift of the elements in the preceding column/row and lastly if each block is a circulant matrix. So, $\boldsymbol{C}$ has the following form*

$$\boldsymbol{C} = \begin{bmatrix} \Delta_0 & \Delta_{n-1} & \cdots & \Delta_1 \\ \Delta_1 & \Delta_0 & \Delta_{n-1} & \cdots \\ \vdots & \ddots & \ddots & \Delta_{n-1} \\ \Delta_{n-1} & \cdots & \Delta_1 & \Delta_0 \end{bmatrix} \tag{2.8.4}$$

where each block Δ_j is an $n_x \times n_x$ circulant matrix.

Definition 11 *The tensor product of a matrix $A \in \mathbb{R}^{m\times n}$ and a matrix $B \in \mathbb{R}^{p\times q}$ is a matrix G of size $(mp) \times (nq)$ which is given by*

$$A \otimes B = G = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2N}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \tag{2.8.5}$$

2.9 Well-posedness

In this section, we give some usual Hilbert spaces with their associated inner products and norms. Moreover, we introduce the notion of the well-posedness. We introduce some operators like Fredholm integral operator and we give the definition of the compact operators. Let H_1 and H_2 denote separable Hilbert spaces with inner products $(\cdot,\cdot)_j$ for $j = 1$ and 2 respectively and norms

$$\| f \|_j = \sqrt{(f,f)_j}, \quad j = 1, 2,$$

where $f \in H_j$. For smooth $f : \mathbb{R}^n \to \mathbb{R}$, define the gradient of f by

$$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}).$$

For a vector valued function $\vec{v} = (v_1, v_2, ..., v_n)$ where each $v_i : \mathbb{R}^n \to \mathbb{R}$ is smooth we define the divergence of $\vec{v}$ by

$$\nabla \cdot \vec{v} = \sum_{i=1}^{i=n} \frac{\partial v_i}{\partial x_i}$$

If $\mathbf{u} \in \mathbb{R}^n$ then the Euclidean norm of $\mathbf{u}$ is defined by

$$\mid \mathbf{u} \mid = \sqrt{\mathbf{u}^* \mathbf{u}}$$

The following are three examples of Hilbert spaces that we will use in subsequent work.

Definition 12 *Let Ω denotes a simply connected, nonempty, measurable set in $\mathbb{R}^n$ that has a piecewise Lipschitz continuous boundary. The Hilbert space $L^2(\Omega)$ consists of all measurable real valued functions f such that $\int_\Omega f(x)^2 dx < \infty$. The L^2 inner product is denoted by*

$$(f, g)_{L^2} = \int_\Omega f(x)g(x)dx, \quad f, g \in L^2.$$

We define the second Hilbert space which is H^1 as follows:

Definition 13 *The H^1 inner product of a pair of smooth functions is given by*

$$(u, v)_{H^1} = \int_\Omega u(x)v(x)dx + \int_\Omega \nabla u(x) \cdot \nabla v(x)dx$$

Now, we need to define what we mean by the well- posedness. Let $\mathbf{K}$ be a mapping from H_1 to H_2.

Definition 14 *The problem*

$$\boldsymbol{K}u = z, \quad u \in H_1 \quad z \in H_2, \tag{2.9.1}$$

is called well- posed if the following conditions are satisfied:

1 *A solution exists, i.e. for any $z \in H_2$, there is $u \in H_1$ such that $\boldsymbol{K}u = z$.*

2 *This solution is unique.*

3 *This solution is stable, that is, given $u^* \in H_1$ and $z^* \in H_2$ for which $\boldsymbol{K}u^* = z^*$ then $\forall\ \epsilon$, $\exists\ \delta(\epsilon) > 0$ such that when $\| z - z^* \|_2 < \delta(\epsilon)$ then $\| u - u^* \|_1 < \epsilon$.*

A problem that is not well posed is called an ill posed problem.

If the mapping $\mathbf{K}$ is linear, the well posedness is equivalent to the requirement that the inverse operator, $\mathbf{K}^{-1} : H_2 \to H_1$ exists and is bounded.

Definition 15 *Let $\boldsymbol{K}$ be a linear operator with a dense domain in H_1 mapping into H_2. The adjoint operator $\boldsymbol{K}^* : H_2 \to H_1$ is a linear operator where for every $y \in \mathcal{D}(\boldsymbol{K}^*)$, there exists a unique $y^* \in H_1$ such that*

$$(\boldsymbol{K}u, y)_2 = (u, y^*)_1, \tag{2.9.2}$$

for every $u \in \mathcal{D}(\boldsymbol{K})$ The adjoint is defined by the mapping $\boldsymbol{K}^* y = y^*$ for all $y \in \mathcal{D}(\boldsymbol{K}^*)$. Where $\mathcal{D}(\boldsymbol{K})$ denotes the domain of the operator $\boldsymbol{K}$.

Definition 16 *The operator $\boldsymbol{K}$ is called compact if the image of any bounded set is relatively compact set. We say that the set $M \subset H_1$ is a relatively compact set if its closure $\overline{M}$ is compact.*

Example 1 *The well known Fredholm integral of the first kind on $L^2(\Omega)$ is an example of a compact operator. Suppose that $k(x,y)$ is measurable function on $\Omega \times \Omega$ and has the following property*

$$\int_\Omega \int_\Omega k(x,y)^2 dxdy < \infty. \tag{2.9.3}$$

Then, the first kind Fredholm integral operator $\boldsymbol{K} : L^2(\Omega) \to L^2(\Omega)$,

$$(\boldsymbol{K}u)(x) = \int_\Omega k(x,y)u(y)dy, \qquad x \in \Omega, \tag{2.9.4}$$

is a compact and the function k is known as the kernel function for the operator $\boldsymbol{K}$.

Definition 17 *Let u be a real valued function on Ω, the total variation (TV) of u is defined by*

$$\mid u \mid_{TV} = \sup_{\vec{w} \in \mathbb{W}} \int_\Omega -u \nabla \cdot \vec{w} dx, \tag{2.9.5}$$

where

$$\mathbb{W} = \{\vec{w} \in \boldsymbol{C}_0^1(\Omega) : \mid \vec{w}(x) \mid \leq 1, \quad \forall x \in \Omega\}, \tag{2.9.6}$$

Definition 18 *The space of functions of bounded variation $BV(\Omega)$ on Ω is the space of all functions u such that $\int_\Omega \mid u \mid dx < \infty$ and $\mid u \mid_{TV} < \infty$.*

Now when $u \in \mathbf{C}^1(\Omega) \cap BV(\Omega)$ then

$$\mid u \mid_{TV} = \sup_{\vec{w} \in \mathbb{W}} \int_\Omega \vec{w} \cdot \nabla u dx. \tag{2.9.7}$$

Moreover, if $| \nabla u | \neq 0$, the supremum of (2.9.7) appears when $\vec{w} = \frac{\nabla u}{|\nabla u|}$. Then

$$| u |_{TV} = \int_\Omega | \nabla u | \, dx. \tag{2.9.8}$$

In the following sections, we give some definitions related to the stochastic pdes.

2.10 Random variables and random fields

Now, we present some fundamental concepts and formulas related to random variables and random fields. First, we define a random variable which is a mathematical tool used to model randomness. We succeed this by introducing some important characteristics associated with random variables, such as the expectation, variance and independence. Then, we define a random field and also the associated mean, variance and the covariance functions. Finally, we introduce the well known (KL) expansion of a random field.

Definition 19 *Consider the probability space $(\Omega, \mathbb{F}, \mathbb{P})$, a function $X : \Omega \to \mathbb{R}$ is called a random variable (r.v.) if $X^{-1}(B) \in \mathbb{F}$ where B is a Borel set in the Borel sigma algebra $\mathbb{B}$.*

Definition 20 *The expected value of X, $\boldsymbol{E}[X]$, is defined by*

$$\boldsymbol{E}[X] := \int_\Omega X(\omega) d\mathbb{P}(\omega) = \int_\mathbb{R} x f(x) dx, \tag{2.10.1}$$

where f is the probability density function (pdf) associated with X.

Definition 21 *The variance of X, $\boldsymbol{Var}[X]$, is defined by*

$$\boldsymbol{Var}[X] := \boldsymbol{E}[X^2] - \boldsymbol{E}[X]^2. \tag{2.10.2}$$

Definition 22 *The n–th moment of X, denoted by $\boldsymbol{E}[X^n]$, is defined as*

$$\boldsymbol{E}[X^n] := \int_{\mathbb{R}} x^n f(x) dx. \tag{2.10.3}$$

where n is a non-negative integer.

Definition 23 *Let X and Y be two random variables with joint density function $f(x,y)$. We say that they are independent if $f(x,y) = f(x)f(y)$ where $f(x)$ and $f(y)$ are the pdfs of X and Y respectively.*

Definition 24 *The set $\{X_m\}$ of random variables is called orthogonal set if $\boldsymbol{E}[X_m X_n] = 0$ holds for all different and positive integers m and n. Moreover, it is called orthonormal set if $\boldsymbol{E}[X_m X_n] = \delta_{mn}$ holds where δ_{mn} is the Kronecker delta function.*

Definition 25 *Let $(\Omega, \mathbb{F}, \mathbb{P})$ be a probability space, a random field*

$$a(.,.) : D \times \Omega \to \mathbb{R}. \tag{2.10.4}$$

is a measurable function from $D \times \Omega$ to $\mathbb{R}$ with respect to the sigma–algebra $\mathbb{F}$ on the sample space Ω and the Borel sigma–algebra on the domains D and $\mathbb{R}$. Here, $D \subset \mathbb{R}^n$ denotes a bounded spatial domain. We assume that for a given random field, its mean and the covariance function are known.

Definition 26 *For a random field a, the mean field is defined as*

$$\boldsymbol{E}_a(x) := \boldsymbol{E}[a(x,.)] = \int_{\Omega} a(x,\omega) d\mathbb{P}(\omega), \tag{2.10.5}$$

and the covariance function as

$$\begin{aligned}\boldsymbol{Cov}_a(x_1;x_2) &= \boldsymbol{E}[(a(x_1,.)-\boldsymbol{E}_a(x_1))(a(x_2,.)-\boldsymbol{E}_a(x_2))]\\ &= \int_\Omega[(a(x_1,\omega)-\boldsymbol{E}_a(x_1))(a(x_2,\omega)-\boldsymbol{E}_a(x_2))]d\mathbb{P}(\omega)\end{aligned} \tag{2.10.6}$$

The variance of the random field a is given by $\boldsymbol{Var}_a(x)=\boldsymbol{Cov}_a(x;x)$. For the mean field and the covariance function to exist in the L^2 -sense, we must require that the random field has a finite second moment, that is, $a\in L^2_{\mathbb{P}}(\Omega;L^2(D))$.

Definition 27 *(Positive semidefinite function). A function $\boldsymbol{V}a\in L^2(D\times D)$ is positive semidefinite on D if*

$$0\le\sum_i^n\sum_j^n c_i\,\boldsymbol{V}a(x_i,x_j)\bar{c}_j, \tag{2.10.7}$$

holds for any positive integer n, for any sequence of complex weights $\{c_i\}_{i=1}^{i=n}$, and for all $x_i,x_j\in D$.

Common covariance functions are of the form

$$\mathbf{Cov}_a=\sigma_a^2\exp(\frac{-|x_1-y_1|}{\tau_1}-\frac{-|x_2-y_2|}{\tau_2}), \tag{2.10.8}$$

$$\mathbf{Cov}_a=\sigma_a^2\exp(\frac{-r}{\tau}), \tag{2.10.9}$$

$$\mathbf{Cov}_a=\sigma_a^2\exp(\frac{-r^2}{\tau^2}), \tag{2.10.10}$$

where r is distance between x and y in the Euclidean norm. The positive constants τ,τ_1 and τ_2 are called the correlation length.

Chapter 3

IMAGE DEBLURRING PROBLEM

3.1 Introduction

Image deblurring problem is one of the most classic linear inverse problems. It is useful technique to make pictures sharp and clear. It is known that a small image often has about $256^2 = 65536$ pixels. There are many sources for blur in images for example: the motion of either the camera or/and object, the environmental effects and the limitations of the optical system.

In these and other situations, the record image has a blur. In image deblurring, we aim to remove this blur and reconstruct a sharp image by using a mathematical model. In the following section, we present the mathematical model behind the image deblurring problems.

3.2 Mathematical model

To deblur an image, we need a mathematical model for how it was blurred. The relation between the true image and blurred image is given by

$$z = \mathbf{K}u + \varepsilon, \tag{3.2.1}$$

where z is the recorded image and u is the original image, $\mathbf{K}$ denotes the blurring operator and ε denotes a noise function. Both blurring and noise affect the quality of the received image. $\mathbf{K}$ is typically a Fredholm integral operator of the first kind(a convolution operator),

$$(\mathbf{K}u)(x) = \int_\Omega k(x, x')u(x')dx', \qquad x \in \Omega \tag{3.2.2}$$

with translational invariance kernel $k(x, x') = k(x - x')$ and Ω is the domain of the image and typically is a square (or rectangle) in $\mathcal{R}^2$ on which the image intensity function u is defined. $x = (\underline{x}, \underline{y})$ denotes the location in Ω. The kernel in (3.2.2) is also called the point spread function (PSF) (see Chapter 2 for some assumptions on the kernel). For Gaussian blurring with parameter σ, the kernel is given by

$$k(x - x') = \frac{1}{2\pi\sigma^2} e^{-\frac{|x-x'|^2}{2\sigma^2}} \tag{3.2.3}$$

There are several kernels given in the literature (see for example [14]). The equation (3.2.1) represents both the deblurring and the denoising problem. If $\varepsilon = 0$, then (3.2.1) is called pure deblurring problem and is called denoising problem when $\mathbf{K} = \mathbf{I}$ where $\mathbf{I}$ is the identity operator. In this research work, we consider the case of pure deblurring problem

$$z = \mathbf{K}u, . \tag{3.2.4}$$

In this case, the problem is to reconstruct u from given data z and blur kernel k. Some times the blur kernels are unknown. In this case the problem is called the blind deconvolution problem (see [102] and [70] for the blind problem). The problem (3.2.4) is an inverse problem. It is known that the operator $\mathbf{K}$ is compact ([1], [98]) (see also Chapter 2 for the compact operators), so problem (3.2.4) is ill-posed (the solution is unstable) and the resulting matrices of discretization are highly ill-conditioned ([1], [98], [51]). In the literature, 'regularization' methods (see [53]) deal with the ill-posedness of the problem. Different approaches use different regularization terms such as Tikhonov regularization and Total Variation regularization and so on (see [4]).

3.2.1 Tikhonov regulazation

Tikhonov regularization is often used to stabilize problem (3.2.4) [96]. In this case, the problem is to find a u which minimizes the functional

$$T(u) = \frac{1}{2} \parallel \mathbf{K}u - z \parallel^2 + \alpha J(u), \tag{3.2.5}$$

with positive parameter α and

$$J(u) = \int_\Omega u^2 dx. \tag{3.2.6}$$

The advantages of the functional (3.2.6) is that it is not difficult to compute. However, the disadvantage is that the reconstructed image includes oscillation or ringing when the recorded image has discontinuity. Another regularization term is [98]

$$J(u) = \int_\Omega \mid \nabla u \mid^2 dx, \tag{3.2.7}$$

where $\nabla(\cdot)$ is the gradient operator and $\mid \cdot \mid$ is the Euclidian norm. Note that the functional (3.2.7) requires u to be smooth. Hence both regularization terms (3.2.6) and (3.2.7) are not suitable when the recorded image has discontinuity or when we need to construct sharp images [98]. Rudin, Osher and Fatemi [83] proposed using Total Variation as a regularization functional.

3.2.2 Total variation(TV)

In the total variation (TV), the regularization functional is defined by

$$J_{TV}(u) =: \int_\Omega \mid \nabla u \mid, \tag{3.2.8}$$

see Definition 18 for the (TV) regularization. In the above functional, u need not to be continuous (see [1]). However, the derivative of the integrand function in equation (3.2.8) does not exist at zero. One remedy of this issue [51] is to add a constant β [98] as follows

$$J_\beta(u) = \int_\Omega \sqrt{\mid \nabla u \mid^2 + \beta^2}. \tag{3.2.9}$$

Then the functional to be minimized is

$$T(u) = \frac{1}{2} \parallel \mathbf{K}u - z \parallel^2 + \alpha \int_\Omega \sqrt{\mid \nabla u \mid^2 + \beta^2}, \tag{3.2.10}$$

with $\alpha,\ \beta > 0$. Under mild conditions on the operator $\mathbf{K}$, the well-posedness of this minimization problem is established in [1]. There are several methods to obtain this minimum given in [4].

3.2.3 The Euler-Lagrange equations

The Euler-Lagrange equations associated with the above minimization problem are [98]:

$$\mathbf{K}^*(\mathbf{K}u - z) + \alpha L(u)u = 0 \qquad x \in \Omega, \tag{3.2.11}$$

$$\frac{\partial u}{\partial n} = 0 \qquad x \in \partial\Omega, \tag{3.2.12}$$

where $\mathbf{K}^*$ is the adjoint of $\mathbf{K}$. The differential operator $L(u)$ is given by

$$L(u)w = -\nabla.(\frac{1}{\sqrt{\mid \nabla u \mid^2 + \beta^2}} \nabla w). \tag{3.2.13}$$

To get the above equations (3.2.11), consider $f(\epsilon) = T(u+\epsilon v)$ as a real valued function where v is an arbitrary function and ϵ is sufficiently small. Now, to find the minimum or maximum values of the real function f we use the standard technique which is find $f'(\epsilon)$ and then take $f' = 0$ at $\epsilon = 0$. Now

$$f(\epsilon) = T(u + \epsilon v) = \frac{1}{2} \parallel \mathbf{K}(u + \epsilon v) - z \parallel^2 + \alpha \int_\Omega \sqrt{\mid \nabla(u + \epsilon v) \mid^2 + \beta^2}, \qquad (3.2.14)$$

Taking the derivative w.r.t. ϵ, using the boundary condition, integrations by parts and writing the integral as inner product we get

$$\frac{df}{d\epsilon} = 0 \Longrightarrow (\mathbf{K}u - z, \mathbf{K}v) + (\alpha L(u)u, v) = 0. \qquad (3.2.15)$$

and hence using the property of the conjugate operator to get

$$\frac{df}{d\epsilon} = 0 \Longrightarrow (\mathbf{K}^*(\mathbf{K}u - z), v) + (\alpha L(u)u, v) = 0. \qquad (3.2.16)$$

This gives

$$(\mathbf{K}^*(\mathbf{K}u - z) + \alpha L(u)u, v) = 0. \qquad (3.2.17)$$

Since v is arbitrary, one can take $v = \mathbf{K}^*(\mathbf{K}u - z) + \alpha L$ to get the result given in (3.2.11).

Note that (3.2.11) is a nonlinear integro-differential equation of elliptic type. Equation (3.2.11) can be expressed as a nonlinear first order system [23]

$$\mathbf{K}^*\mathbf{K}u - \alpha \nabla . \vec{v} = \mathbf{K}^* z, \qquad (3.2.18)$$

$$-\nabla u + \sqrt{\mid \nabla u \mid^2 + \beta^2} \vec{v} = \vec{0}, \qquad (3.2.19)$$

with the dual, or flux, variable

$$\vec{v} = \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta^2}}. \tag{3.2.20}$$

After eliminating the vector $\vec{v}$ from the above equations (3.2.18-3.2.20), one has the primal system

$$(\mathbf{K}^*\mathbf{K} + \alpha L(u))u = \mathbf{K}^* z, \tag{3.2.21}$$

In [98], Vogel and Oman used the Fixed Point Iteration method to linearize the system 3.2.21 by fixing $u = u^{(k)}$ in the square root term given in equation (3.2.19) or (3.2.20) as follows

$$(\mathbf{K}^*\mathbf{K} + \alpha L(u^{(k)}))u^{(k+1)} = \mathbf{K}^* z, \quad k = 0, 1, ... \tag{3.2.22}$$

In this case, $u^{(k+1)}$ is obtained as the solution of the linear integro-differential equation (3.2.22). [4]).

3.2.4 Discretization steps

To discretize (3.2.18) and (3.2.19), we start by dividing the square domain $\Omega = (0,1) \times (0,1)$ into n_x^2 equals squares (cells) where n_x denotes the number of equispaced partitions in the x or y directions. The cell centers are denoted by (x_i, y_j) and given by

$$\begin{aligned} x_i &= (i - \tfrac{1}{2})h \quad i = 1, ..., n_x, \\ y_j &= (j - \tfrac{1}{2})h \quad j = 1, ..., n_x, \end{aligned} \tag{3.2.23}$$

where $h = \frac{1}{n_x}$. The midpoints of cell edges are given by $(x_{i\pm\frac{1}{2}}, y_j)$ and $(x_i, y_{j\pm\frac{1}{2}})$ where

$$\begin{aligned} x_{i\pm\frac{1}{2}} &= x_i \pm \tfrac{h}{2} \quad i = 1, ..., n_x, \\ y_{j\pm\frac{1}{2}} &= y_j \pm \tfrac{h}{2} \quad j = 1, ..., n_x. \end{aligned} \tag{3.2.24}$$

The set

$$e_{ij} = \{(x,y) : x \in [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}],\ y \in [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]\}, \tag{3.2.25}$$

represents a cell with (x_i, y_j) as its center. Let

$$\chi_i(x) = \begin{cases} 1, & \text{if } x \in (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}); \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.26}$$

$$\chi_j(y) \begin{cases} 1, & \text{if } y \in (y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}); \\ 0, & \text{otherwise,} \end{cases} \tag{3.2.27}$$

Approximate u as

$$u(x,y) \simeq U(x,y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} u_{ij} \chi_i(x) \chi_j(y), \tag{3.2.28}$$

where $U(x_i, y_j) = u_{ij}$, and represent the data z as

$$z(x,y) \simeq Z(x,y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} z_{ij} \chi_i(x) \chi_j(y), \tag{3.2.29}$$

where z_{ij} may be calculated as cell averages. Also, approximate v by

$$v(x,y) \simeq \sum_{i=1}^{n_x-1} \sum_{j=1}^{n_x} V_{ij}^x \begin{pmatrix} \phi_i(x)\chi_j(y) \\ 0 \end{pmatrix} + \sum_{i=1}^{n_x-1} \sum_{j=1}^{n_x} V_{ij}^y \begin{pmatrix} 0 \\ \phi_i(y)\chi_j(x), \end{pmatrix} \tag{3.2.30}$$

where ϕ_i are piecewise linear functions characterized by

$$\begin{aligned} \phi_i(x_{l+\frac{1}{2}}) &= \delta_{il}, \\ \phi_j(y_{k+\frac{1}{2}}) &= \delta_{jk}. \end{aligned} \tag{3.2.31}$$

Now, applying Galerkin's method to (3.2.18) and (3.2.19) together with midpoint quadrature for the integral term given in (3.2.2) and cell center finite difference method (CCFDM) for the derivative part given in equation (3.2.19) (see [37] for more details), one obtains the following system

$$K^*{}_h K_h U + \alpha B^T{}_h V = K^*{}_h Z, \tag{3.2.32}$$

$$\alpha B_h U - \alpha D_h{}^{(k)} V = 0. \tag{3.2.33}$$

Here K_h is a matrix of size $n \times n$ and B_h is a matrix of size $m \times n$. $D_h{}^{(k)}$ is a matrix of size $m \times m$ (here $n = n_x^2$ and $m = 2n_x(n_x - 1)$) where k means fixed point iteration for linearizing the nonlinear term inside the square root. For simplicity we eliminate the subscript h. Then one can write

$$\begin{bmatrix} \alpha D^{(k)} & -\alpha B \\ -\alpha B^T & -K^*K \end{bmatrix} \begin{bmatrix} V \\ U \end{bmatrix} = \begin{bmatrix} 0 \\ -K^*Z \end{bmatrix}, \tag{3.2.34}$$

Both K^*K and $L = B^T D^{-1^{(k)}} B$ are symmetric positive semi definite matrices [98]. The matrix K is a BTTB matrix. The matrix D is a diagonal with positive diagonal entries

$$D^{(k)} = \begin{bmatrix} D^x(U^{(k)}) & 0 \\ 0 & D^y(U^{(k)}) \end{bmatrix}, \tag{3.2.35}$$

where D^x and D^y are $(n_x - 1) \times n_x$ and $n_x \times (n_x - 1)$ diagonal matrices, respectively obtained by discretize the expression $\sqrt{\mid \nabla u^{(k)} \mid^2 + \beta^2}$. The matrix B is given by

$$B = \frac{1}{h} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \tag{3.2.36}$$

where the matrices B_1 $(n_x(n_x-1)\times n)$ and B_2 $(n_x(n_x-1)\times n)$ have the following structures

$$B_1 = \begin{bmatrix} -I & I & 0 & 0 & 0 \\ 0 & -I & I & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -I & I \end{bmatrix}, \tag{3.2.37}$$

where I is the identity matrix of size n_x by n_x.

$$B_2 = \begin{bmatrix} E & 0 & 0 & 0 & 0 \\ 0 & E & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & E \end{bmatrix}, \tag{3.2.38}$$

where E $((n_x-1)\times n_x)$ is given by

$$E = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}. \tag{3.2.39}$$

Note that one can eliminate V from (3.2.32) and (3.2.33) to get the following primal system

$$(K^*K+\alpha L)U = K^*Z. \tag{3.2.40}$$

If Tikhonov regularization is used then (3.2.40) becomes

$$(K^*K+\alpha I)U = K^*Z, \tag{3.2.41}$$

where I is the identity matrix of the same size of K. Another generalized saddle point version [73] of (3.2.34) is

$$\begin{bmatrix} I & K \\ -K^* & \alpha L \end{bmatrix} \begin{bmatrix} V \\ U \end{bmatrix} = \begin{bmatrix} Z \\ 0 \end{bmatrix}. \tag{3.2.42}$$

We note that (3.2.40), (3.2.34) and (3.2.42) are are very large systems. The reason of their huge sizes is that for example an image with 256×256 resolution requires solving system of size $256^2 \times 256^2$. Hence, the only choice of linear solver is an iterative method such as a Krylov subspace methods. Unfortunately, these methods are very slow with ill-conditioned linear systems. One technique to overcome this slowness properties is using an appropriate preconditioner (see [92] for preconditioning). We may use the minimal residuals (MINRES) method [77] with suitable preconditioners.

For the system (3.2.40), Vogel and Oman [99] introduced the product preconditioner with approximating the BTTB matrix by a block circulant with circulant block (BCCB) matrix, while Chan et. al [19] introduced a cosine-transform based preconditioner. Donatelli [26] used another solver for this problem with Dirichlet and periodic boundary conditions. The resulting matrices were BTTB and BCCB. He solved the resulting systems by applying a multigrid method and he showed an optimality property with $O(n)$ arithmetic operations where n is the system size. For the system (3.2.41), Donatelli and Hanke [27] introduced an iterative scheme similar to nonstationary iterated Tikhonov regularization. The rapid convergence of their method is determined by an adaptive strategy for selecting the regularization parameters. For the second version of the generalized saddle point problem (3.2.42), NG and Pan [73] developed new preconditioners. These preconditioners are called Hermitian and skew-Hermitian splitting (HSS). They gave a strategy to choose the HSS parameters to force all eigenvalues of the preconditioned matrices to cluster around one and hence, the Krylov subspace method converges very quickly. For more details on iterative methods for image deblurring we refer to [12]. In this dissertation, we consider the preconditioning technique for solving the primal-daual system (3.2.34). This method is presented in the following chapter.

Chapter 4

PRECONDITIONING TECHNIQUE FOR IMAGE DEBLURRING PROBLEM

4.1 Introduction

In this chapter, we consider the preconditioning technique for solving

$$\underbrace{\begin{bmatrix} \alpha D^{(k)} & -\alpha B \\ -\alpha B^T & -K^*K \end{bmatrix}}_{A} \begin{bmatrix} V \\ U \end{bmatrix} = \begin{bmatrix} 0 \\ -K^*Z \end{bmatrix}, \tag{4.1.1}$$

the above system is obtained from discretaizing the Euler Lagrange equations associated with image deblurring problem (see Chapter 3). The coefficient matrix A of this system is of the generalized saddle point form with high condition number and it has a huge size. Hence, we solve this system by using the minimal residual (MINRES) iteration method with using efficient preconditioner.

This preconditioner is of Murphy, Golub and Wathen (MGW) type [72] and it involves a Schur complement of the A which contains a product of a Toeplitz matrix with Toeplitz blocks (BTTB) and its transpose. This product may not be a BTTB. Hence we approximate this product in three approaches. The first approach is based on approximating the BTTB matrix by Strang circulant approximation (see [91], [18]) while in the second approach, we use the optimal circulant approximation for BTTB matrices [22]. The last approach is approximating the product of BTTB and its transpose by a symmetric BTTB [81]. Symmetric BTTB matrices can always be extended to form symmetric BCCB matrices. The benefit of the circulant or BCCB approximation is that the matrix-vector products that involve $n \times n$ matrix can be computed in $O\ (n \log n)$ operations instead of $O\ (n^2)$. This reduction is due to the fast Fourier transform (FFT) and the Convolution theorem. Moreover, all that is needed for computation is the first column of the circulant matrix, which decreases the amount of required storage. We also show that the preconditioned matrices have the clustering behavior of the eigenvalues. Moreover, we present several numerical examples. These numerical examples show the efficiency of the proposed preconditioners.

4.2 The exact preconditioner

Our starting preconditioner for the system (4.1.1) is

$$P = \begin{bmatrix} \alpha\gamma_1 D & 0 \\ 0 & \gamma_2 S \end{bmatrix}, \tag{4.2.1}$$

where $S = (K^*K + \alpha L)$ is the Schur complement of the matrix A. γ_1 and γ_2 are positive parameters which are used to enforce the clustering of the eigenvalues of the preconditioned matrix around one. Hence, the appropriate iterative method is preconditioned MINRES (PMINRES) [77]. More details on preconditioning techniques can be seen in [11], [72] and [16].

4.3 Eigenvalues estimates

In this section we give a bound for the positive and negative eigenvalues of the preconditioned matrix $P^{-1}A$ but before doing that, we start by discussing the number of the negative and positive eigenvalues of the matrix $P^{-1}A$. Note that the preconditioned matrix $P^{-1}A$ is similar to the matrix $P^{-1/2}AP^{-1/2}$. The matrix $P^{-1/2}AP^{-1/2}$ can be decomposed into

$$\begin{bmatrix} I_m & 0 \\ -\sqrt{\frac{\alpha\gamma_1}{\gamma_2}}S^{-1/2}B^*D^{-1/2} & I_n \end{bmatrix} \begin{bmatrix} \frac{1}{\gamma_1}I_m & 0 \\ 0 & -\frac{1}{\gamma_2}I_n \end{bmatrix} \begin{bmatrix} I_m & -\sqrt{\frac{\alpha\gamma_1}{\gamma_2}}D^{-1/2}BS^{-1/2} \\ 0 & I_n \end{bmatrix},$$

where I_m and I_n are the identities matrices of size $m \times m$ and $n \times n$ respectively. The above decomposition is known as the congruence transformations of the matrix $P^{-1/2}AP^{-1/2}$. By Sylvesters law of inertia (page 403 in [47]) , congruence transformations preserve the signs of the eigenvalues [31]. It follows that the number of the positive eigenvalues of $P^{-1}A$ is m and the number of the negatives is n (here $m > n$). Several bounds on the eigenvalues of the generalized saddle point matrix are established in [84, 90] and [5]. Here we use the bounds given in [Theorem 1 in [5] p 4] obtained by Axelsson.

Theorem 3 *The $m + n$ ($\mu_{-n} \leq \mu_{-n+1} \leq ... \leq \mu_{-1} < 0 < \mu_1 \leq \mu_2 \leq ... \leq \mu_m$) eigenvalues of the generalized eigenvalue problem,*

$$\begin{bmatrix} \alpha D & -\alpha B \\ -\alpha B^T & -K^*K \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} \alpha\gamma_1 D & 0 \\ 0 & \gamma_2 S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.3.1}$$

satisfy the following:

$$\mu_i \in \left[\frac{1}{\gamma_1}, \frac{1 + \sqrt{1 + \frac{4\alpha\gamma_1}{\gamma_2}\sigma_m}}{2\gamma_1} \right] \quad i = 1, ..., m, \tag{4.3.2}$$

$$\mu_{-j} \in \left[-\frac{1}{\gamma_2}, -\frac{1}{\gamma_2 + \alpha\gamma_1\tau} \right] \quad j = 1, ..., n, \tag{4.3.3}$$

where γ_1 and γ_2 are positive parameters. σ_m is the maximum eigenvalue of $S^{-1/2}LS^{-1/2}$ and $\tau = \rho(S^{-1/2}LS^{-1/2})$, the spectral radius.

Proof: We start expressing the preconditioned matrix $P^{-1}A$ in a generalized saddle point matrix. $P^{-1}A$ is similar to $P^{\frac{1}{2}}(P^{-1}A)P^{-\frac{1}{2}} = P^{-\frac{1}{2}}AP^{-\frac{1}{2}} =$

$$
\begin{aligned}
&= \begin{bmatrix} \frac{1}{\sqrt{\alpha\gamma_1}}D^{-\frac{1}{2}} & 0 \\ 0 & \frac{1}{\sqrt{\gamma_2}}S^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \alpha D & -\alpha B \\ -\alpha B^T & -K^*K \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\alpha\gamma_1}}D^{-\frac{1}{2}} & 0 \\ 0 & \frac{1}{\sqrt{\gamma_2}}S^{-\frac{1}{2}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\alpha}{\sqrt{\alpha\gamma_1}}D^{\frac{1}{2}} & \frac{-\alpha}{\sqrt{\alpha\gamma_1}}D^{-\frac{1}{2}}B \\ \frac{-\alpha}{\sqrt{\gamma_2}}S^{-\frac{1}{2}}B^T & \frac{-1}{\sqrt{\gamma_2}}S^{-\frac{1}{2}}K^*K \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\alpha\gamma_1}}D^{-\frac{1}{2}} & 0 \\ 0 & \frac{1}{\sqrt{\gamma_2}}S^{-\frac{1}{2}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\gamma_1}I & -\sqrt{\frac{\alpha}{\gamma_1\gamma_2}}D^{-\frac{1}{2}}BS^{-\frac{1}{2}} \\ -\sqrt{\frac{\alpha}{\gamma_1\gamma_2}}S^{-\frac{1}{2}}B^TD^{-\frac{1}{2}} & \frac{-1}{\gamma_2}S^{-\frac{1}{2}}K^*KS^{-\frac{1}{2}} \end{bmatrix} \\
&= \begin{bmatrix} \hat{M} & \hat{B}^* \\ \hat{B} & -\hat{C} \end{bmatrix} = \hat{\mathcal{A}}.
\end{aligned}
$$

Now one can use Theorem (4.1) with the following

$$
\begin{aligned}
&\hat{M} = \frac{1}{\gamma_1}I, && \hat{B} = -\sqrt{\frac{\alpha}{\gamma_1\gamma_2}}S^{-\frac{1}{2}}B^TD^{-\frac{1}{2}}, \\
&\hat{C} = \frac{1}{\gamma_2}S^{-\frac{1}{2}}K^*KS^{-\frac{1}{2}}, && \hat{S} = \frac{1}{\gamma_2}I_n, \\
&\lambda_{max}(\hat{S}) = \frac{1}{\gamma_2}, && \lambda_{min}(\hat{S}) = \frac{1}{\gamma_2}, \\
&\hat{\mu_1} = \frac{1}{\gamma_1}, && \hat{\mu_n} = \frac{1}{\gamma_1}, \\
&\hat{\sigma_m} = \text{maximum eigenvlaue of } \frac{\alpha}{\gamma_2}S^{-\frac{1}{2}}LS^{-\frac{1}{2}}, && \gamma^2 = \rho(\alpha S^{-1/2}LS^{-1/2}),
\end{aligned}
$$

to obtain the bound given in (4.3.2) and (4.3.3).

Remark 1

In the above theorem and its proof, since both P and S are positive definite then $P^{-1/2}$, $P^{1/2}$ and $S^{-1/2}$ are well defined.

Remark 2

If $\gamma_1 = \gamma_2 = 1$, then (4.3.2) and (4.3.3) are given by

$$\mu_i \in \left[1, \frac{1+\sqrt{1+4\alpha\sigma_m}}{2}\right] \quad i = 1, ..., m, \tag{4.3.4}$$

$$\mu_{-j} \in \left[-1, -\frac{1}{1+\alpha\tau}\right] \quad j = 1, ..., n. \tag{4.3.5}$$

Remark 3

From (4.3.2) and (4.3.3), one can note that the smaller value of $\frac{\gamma_1}{\gamma_2}$ yields the smaller length of both intervals. This means that we have a good clustering behavior for the negative and positive eigenvalues. Hence, we expect fast convergence.

4.3.1 Numerical results for the eigenvalues analysis

Our aim is to verify that the bounds given in Theorem (3) are matched with the following numerical example. In this example we take $n = 16$, $\beta = 1$ and $\alpha = 8 \times 10^{-5}$ with the kernel described in (3.2.2). Table 4 .1 shows the upper and lower (positive/negative) bounds of the intervals given in the above lemma. Also it shows the maximum and the minimum (positive/negative) eigenvalues of the preconditioned matrix $P^{-1}A$. These eigenvalues are computed using the built-in Matlab command *eig* (see Chapter 8 for the matlab code). In Table 4 .1, observe that all intervals in the third column are contained in the second column. This observation verifies the bounds given in Theorem (3).

It is known that the PMINRES convergence estimate [31] can be written as

$$\frac{\| r^{(k)} \|_{P^{-1}}}{\| r^{(0)} \|_{P^{-1}}} \leq \underset{q_k \in \Pi_k \ q_k(0)=1}{min} \ \underset{\lambda \in \sigma(P^{-1}A)}{max} \mid q_k(\lambda) \mid, \tag{4.3.6}$$

where Π_k is the space of all polynomial of degree less than or equals k and $\| r^{(0)} \|^2_{P^{-1}} =$

${r^{(0)}}^T P^{-1} r^{(0)}$. To minimize (4.3.6), we need to cluster both the positive and negative eigenvalues. This can be obtained by reducing the lengths of the intervals in (4.3.2) and (4.3.3).

4.4 Approximation K^*K

We introduced some definitions related to the Toeplitz and circulant matrices and their blocks. Now we are ready to speak about the preconditioners of the Toeplitz and BTTB matrices by circulant and BCCB matrices. Circulant preconditioning for Toeplitz systems was introduced by Strang [91] and extended by others to block Toeplitz systems [24]. Many researchers use a Toeplitz preconditioners and block Toeplitz preconditioners for Toeplitz systems see for instance [20] and [63]. Band Toeplitz preconditioner and band BTTB preconditioner are proposed in Chan [18] and Serra [88]. In [64], BTTB preconditioners for BTTB systems are discussed. In our dissertation, we use three approaches to approximate the product K^*K given in the (2,2)-block of the exact preconditioner matrix P.

4.4.1 Symmetric BTTB approximation

Note that our matrix K is a BTTB matrix but the product K^*K need not be BTTB. So, in the first approach, we follow [81] to approximate K^*K given in the preconditioner matrix P by a symmetric BTTB matrix T. Symmetric BTTB matrices can always be extended to form symmetric BCCB matrices. To make the idea clear, we consider the following example

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix} \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & \mathbf{2} \\ 2 & 1 & 2 & \mathbf{3} \\ 3 & 2 & 1 & \mathbf{2} \\ \hline \mathbf{2} & \mathbf{3} & \mathbf{2} & \mathbf{1} \end{array}\right] \tag{4.4.1}$$

This example show how to extend symmetric BTTB matrix into a BCCB. The benefit of this approximation is that the matrix-vector products that involve $n \times n$ matrices can be computed in $O\ (n \log n)$ operations due to the FFT's and the Convolution Theorem. Moreover, all that is needed for computation is the first column of the matrix, which decreases the amount of required storage.

4.4.2 Strang circulant approximation

The second approach that we follow is that we approximate the n by n Toeplitz matrix K given in the preconditioner matrix P by the well known Strang circulant matrix S with diagonals s_k (see [21] page 17–18). In this approximation, if $n = 2m + 1$ the diagonals s_k of S are given by

$$s_k = \begin{cases} k_k, & 0 \leq k \leq m, \\ k_{k-n}, & m < k < n-1, \\ \bar{s}_{-k}, & 0 < -k < n-1, \end{cases} \tag{4.4.2}$$

where k_i is the ith diagonal of the matrix K. If $n = 2m$, we get the Strang matrix S as above. In this case, we define $s_m = 0$ or $s_m = \frac{k_m + k_{-m}}{2}$.

4.4.3 The best circulant approximation

In the last approach, we also approximate the Toeplitz matrix K given in the preconditioner matrix P by an optimal circulant [22]) matrix C. If C_n denote the set of $n \times n$ circulant matrices. The optimal circulant approximation to $K \in \mathbb{C}^{n\times n}$ in the Frobenius norm is given by $C = \arg\min_{B\in C_n} \parallel B - K \parallel_{Fro}$. In this case, the value of the entries c_k of the matrix C is obtained by this formula $c_k = \frac{kT_{-(n-k)}+(n-k)T_k}{n}$, $k = -(n-1), ..., 0, ..., (n-1)$. Resulting of the above three approximations, we have the following three approximation preconditioners.

4.5 Three block diagonal preconditioners

In this section, we introduce the following three block diagonal preconditioners

$$P_T = \begin{bmatrix} \alpha\gamma_1 D & 0 \\ 0 & \gamma_2(T+\alpha L) \end{bmatrix}, \quad P_S = \begin{bmatrix} \alpha\gamma_1 D & 0 \\ 0 & \gamma_2(S^*S+\alpha L) \end{bmatrix},$$
$$P_C = \begin{bmatrix} \alpha\gamma_1 D & 0 \\ 0 & \gamma_2(C^*C+\alpha L) \end{bmatrix}.$$

In the above preconditioners, the matrices T, S and C denote the symmetric BTTB, the Strang BCCB and the best BCCB approximations to the product K^*K given in the exact preconditioner (4.2.1). These approximations allow us to use the FFT the Convolution Theorem. In this case, the matrix-vector products that involve $n \times n$ matrices can be computed in $O\ (n \log n)$ operations. Moreover, all that is needed for the computation is the first column of the matrix, which decreases the amount of required storage.

4.6 Numerical experiments

The aim of this section is to investigate the efficiency of the three preconditioners described above for two blurry images. The first image is a retinal image of a diabetic patient (see Figure 4.2) and the second one is goldhill image (see Figure 4.8). We start by blurring these two images by a certain kernel given in Figure (4.7). Then we deblur these images back and solve the linear system by preconditioned MINRES method using the above three preconditioners (with $\gamma_1 = \gamma_2 = 1$) and we use the well known fixed point iteration method to linearize the non-linear term. We watch the CPU-time and the number of MINRES iterations. It is known that in each PMINRES iteration, we solve a linear system of the form $Px = y$. To solve this system, we use the conjugate gradient method (CG) for the (2,2) block.

Example 2 *In this example, we calculate the iterations number of MINRES with using the three preconditioners P_T, P_S and P_C. We fix the maximum iteration of PMINRES to be* 100*, the tolerance* $1e-2$*,* $\beta = 0.01$*,* $\alpha = 0.00008$*, and we use the retinal image (blurred image) given in Figure (4.3) as a data with PSNR* $= 20.5548$*.* ***Firstly,*** *we start by using the preconditioner P_T. Table (4 .2) shows the degree of freedom (dof), the PMINRES iterations and the PSNR in each iteration of the fixed point method.*

Secondly, *we use preconditioner P_S with the same blurred image and the same parameters given above. Table (4 .3) shows the degree of freedom (dof), the PMINRES iterations and the PSNR in each iteration of the fixed point method.*

Finally, *we use preconditioner P_C with the same bulurred image and the same parameters given above. Table (4 .4) show the degree of freedom (dof), the PMINRES iterations and the PSNR in each iteration of the fixed point method. For the qualities of the reconstruction images using these three preconditioners, see Figures (4.4-4.6).*

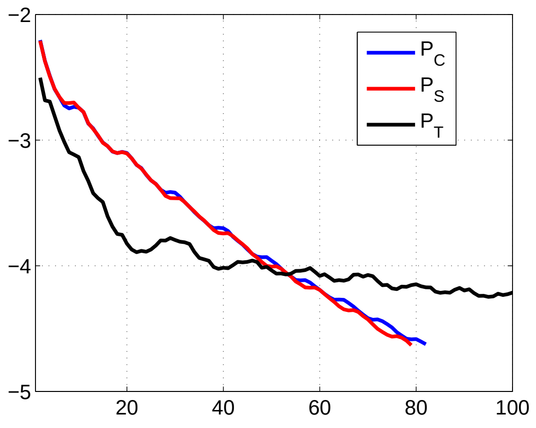


Figure 4 .1: Iterations Number v.s. the Residual

In this example, the second computations carried out for the second data (blurred image) given in Figure (4.9) which is blurred by the kernel given in Figure (4.13). The qualities of the reconstruction images are shown in Figures (4.10-4.12).

Example 3 *In this example we compare the CPU-time of the three PMINRES preconditioned. In Table (4 .5), we list the CPU-time of the PMINRES spends to do 5 fixed point iterations.*

Example 4 *In this example, we compute the residual of PMINRES using the three preconditioners with the same bulurred image and the same parameters given in the above examples. Figure (4.1) shows the convergence of the methods. From Figure (4.1), it can be seen that the preconditioner P_S is the fastest one followed by P_C and then P_T. It is clear that P_S needs 78 iterations to reach the tol $= 1\mathrm{e}-2$, P_C needs 81 while P_T needs more than 100 iterations to reach the same tolerance. Note that we take the PMINRES iterations for these three preconditioners at the second iteration of the fixed point iteration method.*

Example 5 *In this example, we use the true image given in Figure 4.8 and the* *blurred images given in Figure 4.9, (it is blurred by using the kernel given in Figure 4.13), and we fix the preconditioner to be P_T. We watch the quality of the deblurred images in some fixed point iteration. Figures (4.14-4.17) show the deblurred images in the iterations number: 1, 5, 10 and 13. The second computations carried out for different values of the regularization parameters α. Figures (4.18-4.21) show the deblurred images for $\alpha = 8e-2, 8e-4, 8e-7, 8e-8$.*

Remark 4

In all the above examples (3-6), we fix $\gamma_1 = \gamma_2 = 1$. In the following example, we change the values of these two parameters to show how do they affect the convergence of the MINRES method. For this test, we just consider the preconditioner P_T and we vary the values of the parameters.

Example 6 *In this example, we have chosen $n_x = 128$ and $\beta = 0.01$. Here P_0 refers to no-preconditioner, P_{AN} to P_T with $\gamma_1 = \gamma_2 = 1$, P_{12} to P_T with $\gamma_1 = 1,\ \gamma_2 = 10$, P_2 to P_T with $\gamma_1 = 1e-3,\ \gamma_2 = 1$ and finally P_3 refers to P_T with $\gamma_1 = 1e-6,\ \gamma_2 = 1$. In Figures (4.22) and (4.23) observe that unpreconditioned MINRES converged most slowly, followed by PMINRES P_{AN} and then both P_0 and P_{AN} are followed by P_{12}. We note that PMINRES P_3 is the fastest one. This has the smallest value of the parameter γ_1 which leads to the best clustering behavior of the eigenvalues (see Remark 3 and Table 4.1). Figures (4.24-4.25) show the difference between the unpreconditioned MINRES (P_0) and PMINRES P_{AN}.*

Finally, the CPU time and the measure of image quality, Peak Signal-to-Noise Ratio (PSNR), for the preconditioners P_{AN}, P_{12}, P_2 and P_3 are given in Table 4 .6. In this table, we compute the CPU time for 15 iterations for P_{AN} to reach $tol = 1e-3$, 10 iterations for P_{12} to reach $tol = 1e-3$, 7 iterations for P_2 to reach $tol = 1e-3$ and 6 *iterations for P_3 to reach the same tolerance. Through this comparison, we find that the PSNR for the blurred image is (21.2004) while the PSNR for deblurred image can be seen in Table 4 .6.*

Remark 5

P_{AN} denotes the Axelsson and Neytcheva preconditioner [6] (the exact preconditioner (4.2.1) with $\gamma_1 = \gamma_2 = 1$).

Remark 6

PSNR is defined by:

$$PSNR(u,v) = 10\log_{10}(max(max(u), max(v))^2/|u-v|^2). \tag{4.6.1}$$

Remark 7

All required Matlab-codes for the all above computations can be found in the last chapter.

4.7 Conclusion

Three different preconditioners for the generalized saddle point system resulted from discretizing the Euler Lagrange equations associated with image debulrring problem are presented. In these preconditioners, three approximations for the product of the BTTB matrix and its transpose are considered. From the computations, we observe that the P_S preconditioner is the most effective one followed by P_C and then by P_T.

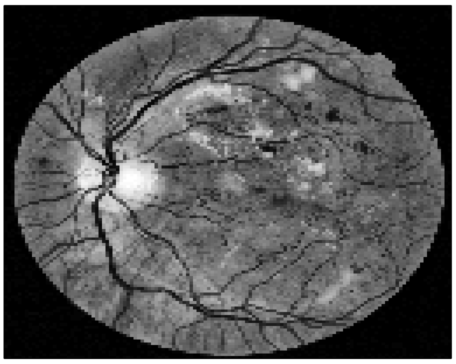


Figure 4 .2: True Image

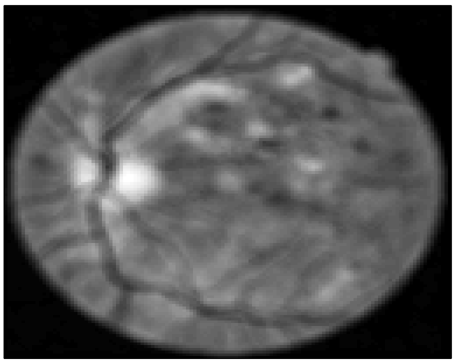


Figure 4 .3: Blurred Image

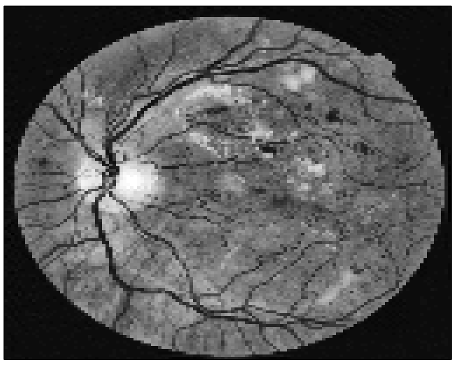


Figure 4 .4: Deblured Image P_T

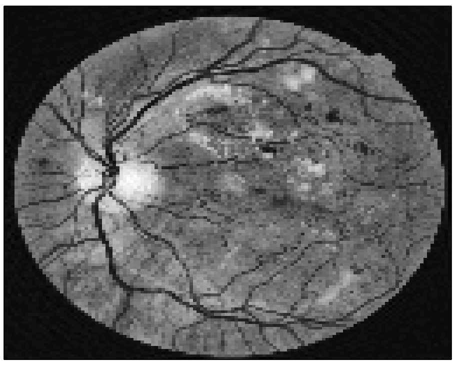


Figure 4 .5: Deblured Image P_S

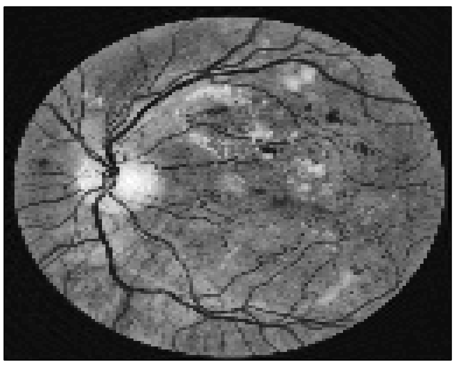


Figure 4 .6: Deblured Image P_C

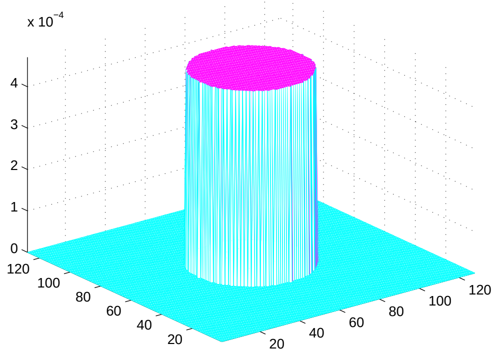


Figure 4 .7: Out-of-focus kernel

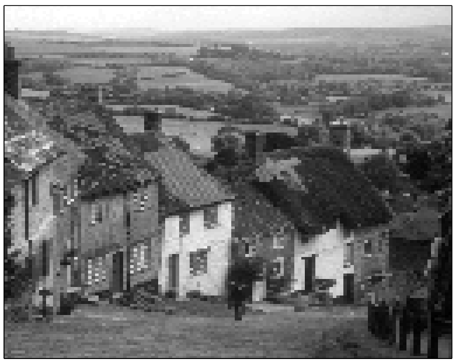


Figure 4 .8: True Image

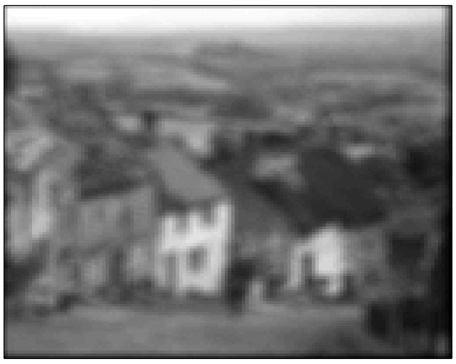


Figure 4 .9: Blurred Image

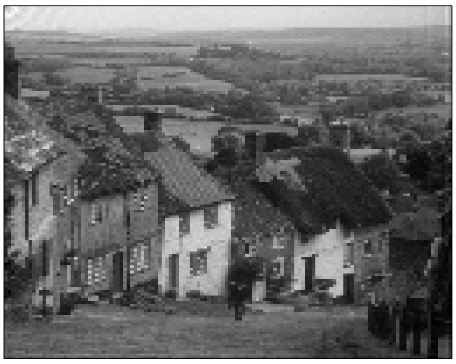


Figure 4 .10: Deblured Image P_T

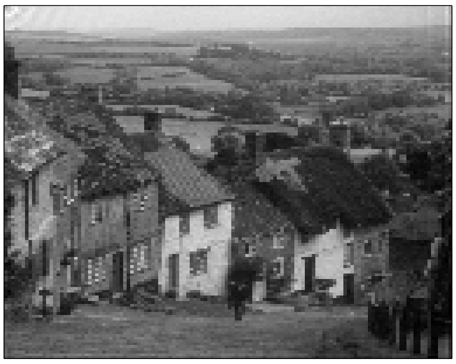


Figure 4 .11: Deblured Image P_S

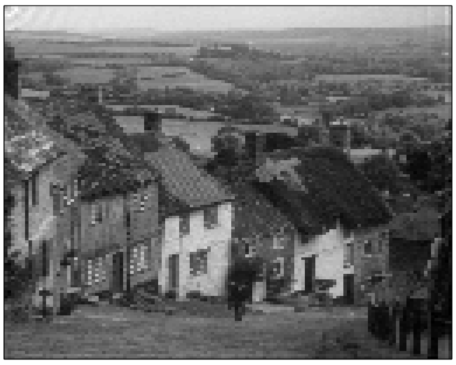


Figure 4 .12: Deblured Image P_C

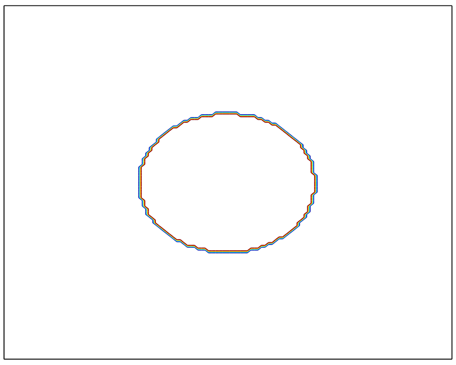


Figure 4 .13: Kernel Cantor

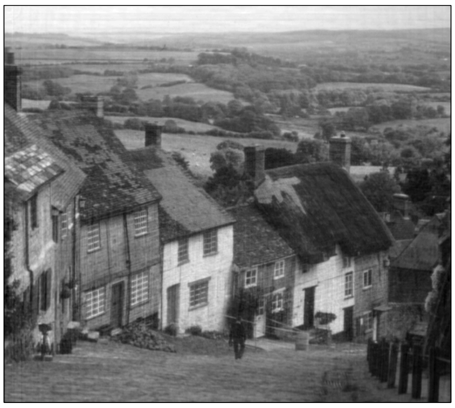


Figure 4 .14: 1st Fixed Point Iteration

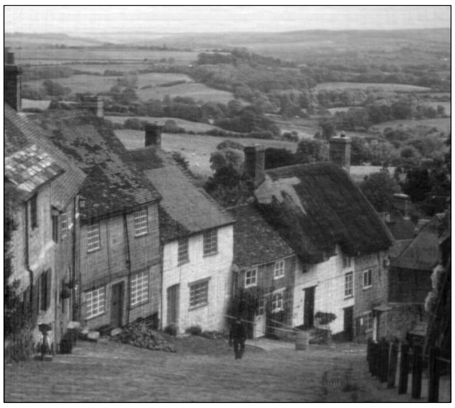


Figure 4 .15: 5th Fixed Point Iteration

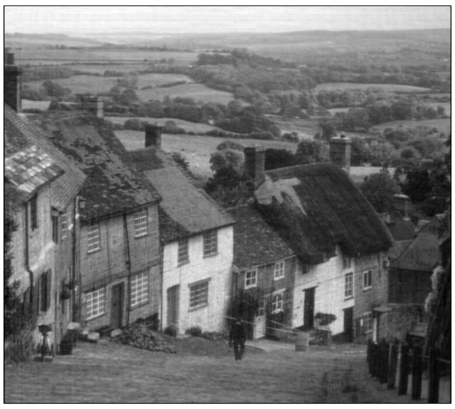


Figure 4 .16: 10th Fixed Point Iteration

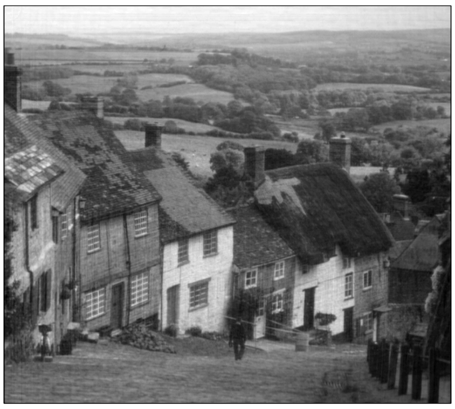


Figure 4 .17: 13th Fixed Point Iteration

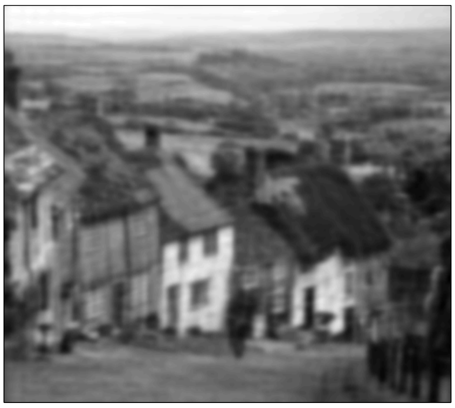


Figure 4 .18: $\alpha = 8.0e-2$

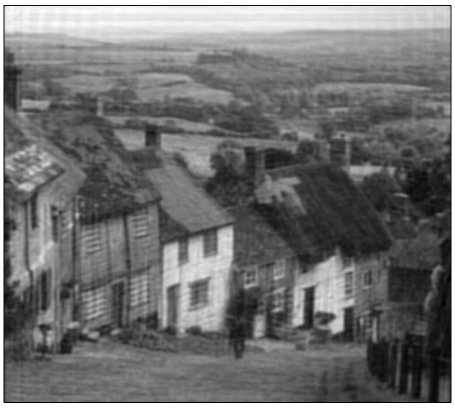


Figure 4 .19: $\alpha = 8.0e-4$

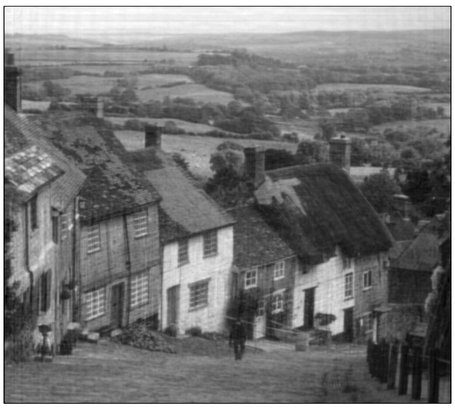


Figure 4 .20: $\alpha = 8.0e-7$

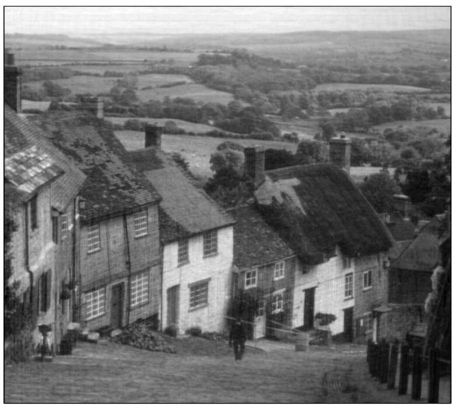


Figure 4 .21: $\alpha = 8.0e-8$

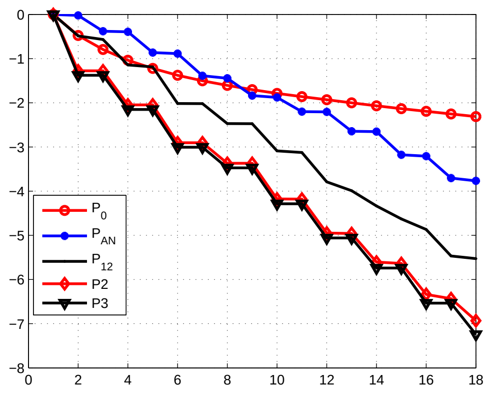


Figure 4 .22: Res. .vs. iter. $\alpha = 8e - 5$

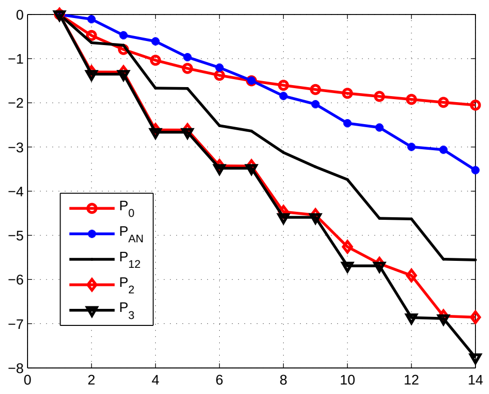


Figure 4 .23: Res. .vs. iter. $\alpha = 8e - 4$

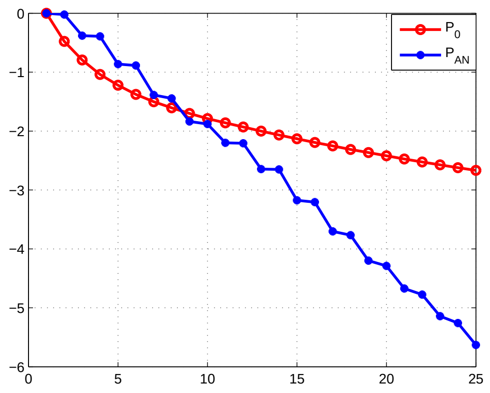


Figure 4 .24: Res. .vs. iter. $\alpha = 8e - 5$

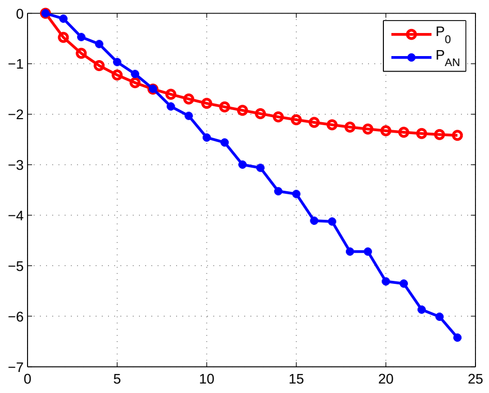


Figure 4 .25: Res. .vs. iter. $\alpha = 8e - 4$

γ_1, γ_2	Bounds in (4.3.2)-(4.3.3)	Computed eigenvalues
$1, 1$	$[-1, -6.42e-1] \cup [1, 1.39]$	$[-1, -7.59e-1] \cup [1, 1.31]$
$1e-3, 1$	$[-1, -9.99444e-1] \cup [1e+3, 1.0005555e+3]$	$[-1, -9.99445e-1] \cup [1e+3, 1.0005552e+3]$
$1e-6, 1$	$[-1, -9.999994441e-1]$ $\cup$ $[1e+6, 1.0000005558257e+6]$	$[-1, -9.999994442e-1]$ $\cup$ $[1e+6, 1.0000005558255e+6]$

Table 4 .1: Bounds on eigenvalues of the preconditioned matrix $P^{-1}A$

Table 4 .2: The Preconditioner P_T

Fixed Point Iteration Number	nx	dof	PMINRES Iteration	PSNR
1	128	48896	> 100	40.6813
2	128	48896	> 100	42.2709
3	128	48896	14	42.5842
4	128	48896	3	42.5841
5	128	48896	1	42.5841

Table 4 .3: The Preconditioner P_S

Fixed Point Iteration Number	nx	dof	PMINRES Iteration	PSNR
1	128	48896	> 100	40.6510
2	128	48896	78	42.6645
3	128	48896	6	42.6688
4	128	48896	1	42.6688
5	128	48896	1	42.6688

Table 4 .4: The Preconditioner P_C

Fixed Point Iteration Number	nx	dof	PMINRES Iteration	PSNR
1	128	48896	> 100	40.6493
2	128	48896	81	42.6535
3	128	48896	6	42.6577
4	128	48896	1	42.6577
5	128	48896	1	42.6577

Table 4 .5: Comparison between P_T, P_S and P_C

nx	dof	CPU Time of P_T	CPU Time of P_S	CPU Time of P_C
128	48896	74.706	39.243	42.653

Table 4 .6: CPU time, PSNR for P_{AN}, P_{12}, P_2 and P_3

	P_{AN}	P_{12}	P_2	P_3
CPU(in second)	23.59	14.52	12.53	11.24
PSNR for deblurred image (in decibels)	26.6606	26.6673	26.6609	26.6609

Chapter 5

STOCHASTIC DARCY'S EQUATIONS

5.1 Introduction

The flow of a fluid in porous media is described by Darcy's law as follows

$$K^{-1}(x)\vec{u}(x) - \nabla p(x) = 0, \tag{5.1.1}$$

with the divergence constraint

$$-\nabla \cdot \vec{u}(x) = f(x) \quad in \ D, \tag{5.1.2}$$

and the boundary condition

$$\vec{n} \cdot \vec{u}(x) = 0 \quad on \ \ \partial D, \tag{5.1.3}$$

where $K^{-1}(x)$ is the permeability, assumed to be uniformly positive definite and bounded, and f is a given data defined on D satisfying the compatibility condition:

$$\int_D f(x)dx = 0. \tag{5.1.4}$$

In the above equations, $D \subset \mathbb{R}^2$ is a bounded, simply connected, polygonal domain in $\mathbb{R}^2$ whose boundary is ∂D and $\vec{n}$ is the outward normal to the boundary. The vector $\vec{u} : D \to \mathbb{R}^2$ is the velocity and the function $p : D \to \mathbb{R}$ is the pressure. The equations (5.1.1-5.1.4) represent a simple model for a single–phase flow in a porous medium. In some situations, in engineering applications, the coefficient K^{-1} in (5.1.1) is not known at all points of D. As a usual technique, one can consider K^{-1} as a random field. At every $x \in D$, it can be considered as a random variable. To this end, let $(\Omega, \mathbb{F}, \mathbb{P})$ to a complete probability space where $\mathbb{F}$ is the σ-algebra over the sample space Ω and $\mathbb{P} : \mathbb{F} \to [0,1]$ is the probability measure with $\mathbb{P}(\Omega) = 1$. Now, if $K^{-1} = K^{-1}(x,\omega)$, $x \in D$, $\omega \in \Omega$ the solution to (5.1.1-5.1.4) is a pair of random fields $(\vec{u}, p) = (\vec{u}(x,\omega), p(x,\omega))$ such that, $P - a.e.$ in Ω,

$$\begin{aligned} K^{-1}(x,\omega)\vec{u}(x,\omega) - \nabla p(x,\omega) &= 0 \qquad && in\ D \times \Omega, \\ \nabla.\vec{u}(x,\omega) &= -f(x) && in\ D \times \Omega, \\ \vec{n}.\vec{u}(x,\omega) &= 0 && on\ \partial D \times \Omega, \end{aligned} \tag{5.1.5}$$

5.2 Overview

In this chapter, we aim to approximate efficiently the statistical moments (mean and variance) of the unknown (pressure and velocity) given in the mixed problem (5.1.5) via stochastic Galerkin finite element method. Mixed problem (5.1.5) is covered in some papers, see for example ([13], [33], [80], [93], [30], [48], [29], [41]).
In [13], the well-posedness, the regularity of solutions and a priori error estimates for stochastic Galerkin finite element approximations are discussed. In [33], an efficient linear solver for the stochastic Galerkin mixed problem (5.1.5) is presented. In both [13] and [33] stochastic Galerkin discretizations is carried out in the case when the random coefficient is assumed uniformly bounded. In [80], the multilevel Monte Carlo algorithm is used. However, the random coefficient is assumed to be a lognormal random field. In [93], efficient iterative methods for the same problem is studied but when the Gaussian random fields are transformed into lognormal ones.

In our dissertation, we follow the works given in [13, 33] to approximate the statistical moments of the mixed problem (5.1.5) by using stochastic Galerkin finite element method .

The main difference between the approaches in [13, 33] and our approach is that in [13, 33], the mixed problem (5.1.5) are studied and analyzed in term of $H(div(D)) \otimes L^2_{\mathbb{P}}(\Omega)$ and $L^2(D) \otimes L^2_{\mathbb{P}}(\Omega)$ formulation for the spatial velocity and the pressure spaces. This approach leads to a saddle point systems in which the (1,1)-block matrix dose not have a diagonal structure. Hence, the computing may not be easy and the cost of the computation will be high.

However, our approach is based on the $[L^2(D)]^2 \otimes L^2_{\mathbb{P}}(\Omega)$ and $(H^1(D) \cap L^2_0(D)) \otimes L^2_{\mathbb{P}}(\Omega)$ spaces of the velocity and the pressure, respectively, which is the mean contribution in this chapter.

One of the main advantage of our formulation is that the deterministic mass matrix is a diagonal which causes the (1,1)-block matrix in the coefficient matrix of the saddle linear system to have a diagonal structure. The reason for the diagonal property is that the finite dimensional subspace of the velocity space spans by orthonormal elements.

Our new stochastic formulation is an extension of the original work introduced in [2]. In [2], the deterministic version of the mixed problem (5.1.5) has been studied and the error, existence and uniqueness are discussed.

The rest of this chapter is organized as follows: some spaces and their norms are presented in Section 5.3. Weak formulations with their analysis are discussed in Section 5.4. Sections 5.5 contains the approximation of the deterministic spaces. Finite-dimensional noise is studied in Section 5.6. In Section 5.7 the approximation of the stochastic spaces are discussed. Section 5.8 includes the stochastic matrices structure.

5.3 Hilbert spaces

We introduce some usual Hilbert spaces with their associated inner products and norms. Let $L^2(D)$ denotes the Lebesgue square integrable functions with inner-product $(.,.)$ and with norm $\| \, . \, \|_0$. $H^1(D)$ denote the Sobolev space consisting of functions, which together with their distributional derivatives of order one are in $L^2(D)$. The associated inner-product is defined as $(\vec{u}, \vec{v})_{H^1} = (\vec{u}, \vec{v}) + (\nabla\vec{u}, \nabla\vec{v})$ and the norm on $H^1(D)$ is denoted by $\| \, . \, \|_1$. Let $[L^2(D)]^2$ be the space $L^2(D) \times L^2(D)$ whose inner product is understood to hold componentwise and its norm is also denoted by $\| \, . \, \|_0$. More details for the above spaces can be found in [3]. In this paper, we set

$$\begin{aligned} X &= [L^2(D)]^2, \\ Q &= H^1(D) \cap L^2_0(D), \end{aligned} \tag{5.3.1}$$

where $L^2_0(D)$ stands for the space $L^2_0(D) = \{q \in L^2(D) : \int_D q(x)dx = 0\}$. We define

$$\begin{aligned} \| \, \vec{v} \, \|^2_{L^2(D)} &= \int_D \vec{v} \cdot \vec{v} dx, \\ \| \, \vec{v} \, \|^2_X &= \int_D (v_1^2 + v_2^2) dx = \| \, v_1 \, \|^2_{L^2} + \| \, v_2 \, \|^2_{L^2}, \\ \| \, q \, \|^2_{H^1} &= \int_D q^2 dx + \int_D (\nabla q \cdot \nabla q) dx = \| \, q \, \|^2_{L^2} + | \, q \, |^2_{H^1} \, . \end{aligned} \tag{5.3.2}$$

The space $L^2_{\mathbb{P}}(\Omega)$ consists of all random variables with finite second moment i.e. $\mathbf{E}[\xi^2] < \infty$. The inner product of the space $L^2_{\mathbb{P}}(\Omega)$ is defined by $(\xi_1, \xi_2) = \mathbf{E}[\xi_1.\xi_2]$ and the norm $\| \, \xi \, \|_{L^2(\mathbb{P})} = (\xi, \xi)^{\frac{1}{2}}$. For more details of the theory of random fields and stochastic concepts we refer to [50] and [75]. Since the stochastic functions have different structures with x and with ω, we introduce the tensor product spaces as follows, (see [94] for a definition of the tensor product),

$$\begin{aligned} \mathbb{X} &:= X \otimes L^2_{\mathbb{P}}(\Omega), \\ \mathbb{Q} &:= Q \otimes L^2_{\mathbb{P}}(\Omega). \end{aligned} \tag{5.3.3}$$

For $\vec{v}(x,\omega) \in \mathbb{X}$ and $q(x,\omega) \in \mathbb{Q}$, the associated norms are defined by $\| \vec{v} \|^2_{\mathbb{X}} := \langle \| \vec{v} \|^2_X \rangle$ and $\| q \|^2_{\mathbb{Q}} := \langle \| q \|^2_{H^1} \rangle$.

5.4 Weak formulations

To formulate the weak formulation of our problem (5.1.5) it is better to start with the following mixed deterministic Darcy's equations

$$\begin{aligned} K^{-1}(x)\vec{u}(x) - \nabla p(x) &= 0 \qquad \textit{in } D, \\ -\nabla \cdot \vec{u}(x) &= f(x) \qquad \textit{in } D, \\ \vec{n} \cdot \vec{u}(x) &= 0 \qquad \textit{on } \partial D, \end{aligned} \tag{5.4.1}$$

where $K^{-1}(x)$ is assumed to be a 2×2 bounded and spd matrix-valued function. This means that

$$k_{min}(\vec{\gamma},\vec{\gamma}) \leq (K^{-1}\vec{\gamma},\vec{\gamma}) \leq k_{max}(\vec{\gamma},\vec{\gamma}), \tag{5.4.2}$$

where k_{min} and k_{max} are positive constants and for every $\vec{\gamma} : D \to \mathbb{R}^2$.

5.4.1 The weak formulation for the deterministic problem

The weak formulation of (5.4.1) is to find $(\vec{u}, p) \in X \times Q$ such that

$$\begin{aligned} \mathbf{a}(\vec{u}, \vec{v}) + \mathbf{b}(\vec{v}, p) = 0 \qquad & \vec{v} \in X, \\ \mathbf{b}(\vec{u}, w) = \mathbf{l}(w) \qquad & w \in Q, \end{aligned} \tag{5.4.3}$$

where $\mathbf{a}(.,.)$ and $\mathbf{b}(.,.)$ are bilinear forms defined by $\mathbf{a}(\vec{u}, \vec{v}) = \int_D K^{-1}\vec{u} \cdot \vec{v}dx$ and $\mathbf{b}(\vec{u}, w) = -\int_D \vec{u} \cdot \nabla w dx$ The linear form $\mathbf{l}(.)$ is given by $\mathbf{l}(w) = -\int_D fwdx$. To get the above formulation, we multiply the first equation in (5.4.1) by the vector-function $\vec{v} \in X$ and the second one by the scalar-function $w \in Q$ and integrate the second one using the Green formula and the boundary condition.

Theorem 4 *For any data $f \in L^2(D)$, the problem (5.4.3) has a unique solution (u, p) in $X \times Q$ provided that the condition (5.4.2) holds. Moreover this solution satisfies the following estimate*

$$\| \vec{u} \|_{[L^2(D)]^2} + \| p \|_{H^1(D)} \leq k \| f \|_{L^2(D)}, \tag{5.4.4}$$

where k is a constant depend on the constants $(k_{max}, k_{min}, \beta)$ where k_{max}, k_{min} are given in (5.4.2) and β is a positive constant and is the inf-sup constant of (5.4.3).

Proof: The theorem above can be proved using the theory given in [15], [39] and [45]. To follow them, we need to verify the continuity (boundedness) of both bilinear forms $\mathbf{a}(.,.)$ and $\mathbf{b}(.,.)$, the coercivity of $\mathbf{a}(.,.)$ on the null-space (also called kernel space) $Z = \{\vec{v} \in X : \mathbf{b}(\vec{v}, w) = 0 \ \ \forall w \in Q\}$ and the inf-sup condition. The continuity of $\mathbf{a}(.,.)$ on $X \times X$ can be proved using the right inequality of (5.4.2) and using Cauchy Schwarz inequality. It is clear that $\mathbf{a}(.,.)$ is coercive on the null-space Z with using the left inequality of (5.4.2) and the definition of $\| . \|_{[L^2(D)]^2}$. For the continuity of $\mathbf{b}(.,.)$

on $X \times Q$, one can achieved it by using the Cauchy Schwarz inequality. Moreover, for any $w \in Q$, by taking $\vec{r}_q = \nabla w$, we obtain

$$| \mathbf{b}(\vec{r}_q, w) |= \int_D \vec{r}_q \cdot \nabla w = \int_D \nabla w \cdot \nabla w = \int_D |\nabla w|^2 =| w |_{H^1}^2 =\| \vec{r}_q \|_{[L^2(D)]^2} | w |_{H^1(D)} .$$

By using the so called Poincaré inequality on Q, (see [45], Chap. I, Thm 1.9), we obtain the inf-sup condition

$$\forall\ w \in Q, \quad \sup_{\vec{r} \in X} \frac{| \mathbf{b}(\vec{r}, w) |}{\| \vec{r} \|_{[L^2(D)]^2}} \geq \beta \| w \|_{H^1(D)}, \tag{5.4.5}$$

To prove the second part of the theorem, let $v = u$ in the first equation of (5.4.3) to get

$$\mathbf{a}(\vec{u}, \vec{u}) = -\mathbf{b}(\vec{u}, p) = (f, p) \tag{5.4.6}$$

using the coercivity of $\mathbf{a}(.,.)$ and the continuity of $\mathbf{l}(p)$ one gets

$$\| u \|_X^2 \leq C_1 \| f \|_{L^2} \| p \|_Q \tag{5.4.7}$$

and using the inf-sup condition, we have

$$\beta \| p \|_{H^1(D)} \leq \sup_{\vec{v} \in X} \frac{| \mathbf{b}(\vec{v}, p) |}{\| \vec{v} \|_{[L^2(D)]^2}} = \sup_{\vec{v} \in X} \frac{| -\mathbf{a}(\vec{u}, \vec{v}) |}{\| \vec{v} \|_{[L^2(D)]^2}} \leq C_2 \| \vec{u} \|_X \tag{5.4.8}$$

So, equation (5.4.4) can be achieved by substituting (5.4.8) in (5.4.7). The above theorem and its prove can be found in [2] when the $K^{-1}(x) \equiv 1$.

5.4.2 The weak formulation of the stochastic problem

In this subsection, we find the weak formulation of the stochastic problem. We start by assuming the following:

Assumption 1 *$K^{-1}(x, \omega)$ is a second-order random field, in other word, $K^{-1}(x, \cdot) \in L^2_{\mathbb{P}}(\Omega), \ \forall x \in D$*

Assumption 2 *$K^{-1}(x, \omega) \in L^\infty(D \times \Omega)$ satisfies the following*

$$0 < C_{min} \leq K^{-1}(x, \omega) \leq C_{max} < \infty \quad \textit{a.e. in } D \times \Omega. \tag{5.4.9}$$

where C_{max} and C_{min} are positive constants.

Now it is easy to get the weak formulation of the problem (5.1.5) which is to find $\vec{u} \in \mathbb{X}$ and $p \in \mathbb{Q}$ such that

$$\begin{aligned} a(\vec{u}, \vec{v}) + b(\vec{v}, p) &= 0; & \vec{v} \in \mathbb{X}, \\ b(\vec{u}, w) &= -\langle (f, w) \rangle; & w \in \mathbb{Q}, \end{aligned} \tag{5.4.10}$$

where the bilinears $a(.,.)$ and $b(.,.)$ are given by $a(\vec{u}, \vec{v}) = \mathbf{E}[\int_D K^{-1} \vec{u} \cdot \vec{v} dx]$ and $b(\vec{u}, w) = -\mathbf{E}[\int_D \vec{u} \cdot \nabla w dx]$.

Theorem 5 *Let $f \in L^2(D)$, the problem (5.4.10) has a unique solution (u, p) in $\mathbb{X} \times \mathbb{Q}$ provided that the Assumption 2 holds. Moreover the following estimate holds*

$$\| \vec{u} \|_{[L^2(D)]^2 \otimes L^2_{\mathbb{P}}(\Omega)} + \| p \|_{H^1(D) \otimes L^2_{\mathbb{P}}(\Omega)} \leq C \| f \|_{L^2(D)}, \tag{5.4.11}$$

where the positive constant C depends on the constants $(C_{max}, C_{min}, \beta)$ where C_{max}, C_{min} are given in (5.4.9) and β is the inf-sup constant of the problem (5.4.10) and so is of *the problem (5.4.3).*

Proof: The above theorem can be derived as we did in section 4.1. The continuity of $a(.,.)$ on $\mathbb{X} \times \mathbb{X}$ can be achieved using the right inequality of (5.4.9) and using Cauchy Schwarz inequality. The coercivity of $a(.,.)$ on the null-space $\mathbb{Z} = \{\vec{v} \in \mathbb{X} : b(\vec{v}, w) = 0 \;\; \forall w \in \mathbb{Q}\}$ can be achieved by using the left inequality of (5.4.9) and the definition of $\| \, . \, \|_{[L^2(D)]^2 \otimes L^2_{\mathbb{P}}(\Omega)}$. One can show that $b(.,.)$ is continuous on $\mathbb{X} \times \mathbb{Q}$ by using the Cauchy Schwarz inequality. Moreover, for any $w \in \mathbb{Q}$, by taking $\vec{r_q} = \nabla w$, we obtain

$$| \, b(\vec{r}_q, w) \, | = \int_\Omega \int_D \vec{r}_q \cdot \nabla w dx d\mathbb{P} = | \, w \, |^2_{H^1 \times L^2_{\mathbb{P}}} = \| \, \vec{r}_q \, \|_{[L^2]^2 \otimes L^2_{\mathbb{P}}} | \, w \, |_{H^1 \otimes L^2_{\mathbb{P}}} \, .$$

By using the Poincaré inequality on $\mathbb{Q}$, we get the inf-sup condition

$$\forall \; w \in \mathbb{Q}, \quad \underset{\vec{r} \in \mathbb{X}}{sup} \frac{| \, b(\vec{r}, w) \, |}{\| \, \vec{r} \, \|_{[L^2]^2 \otimes L^2_{\mathbb{P}}}} \geq \beta \, \| \, w \, \|_{H^1 \otimes L^2_{\mathbb{P}}}, \tag{5.4.12}$$

where β is a positive constant. To prove the bound (5.4.11), we have

$$a(\vec{u}, \vec{u}) = -b(\vec{u}, p) = (f, p).$$

Using the coercivity of $a(\cdot, \cdot)$ and the Cauchy-Schwartz inequality, we have

$$\| \, \vec{u} \, \|^2_{\mathbb{X}} \leq \frac{1}{C_{min}} \, \| \, f \, \|_{L^2} \| \, p \, \|_{\mathbb{Q}} \, . \tag{5.4.13}$$

From the inf-sup condition, we have

$$\begin{aligned} \| p \|_{\mathbb{Q}} &\leq \frac{1}{\beta} \sup_{\vec{v}\in\mathbb{X}} \frac{| b(\vec{v},p) |}{\| \vec{v} \|_{\mathbb{X}}} = \frac{1}{\beta} \sup_{\vec{v}\in\mathbb{X}} \frac{| -a(\vec{u},\vec{v}) |}{\| \vec{v} \|_{\mathbb{X}}}, \\ \| p \|_{\mathbb{Q}} &\leq \frac{C_{max}}{\beta} \sup_{\vec{v}\in\mathbb{X}} \frac{\| \vec{u} \|_{\mathbb{X}} \| \vec{v} \|_{\mathbb{X}}}{\| \vec{v} \|_{\mathbb{X}}} = \frac{C_{max}}{\beta} \| \vec{u} \|_{\mathbb{X}} . \end{aligned} \tag{5.4.14}$$

Using the above equation in (5.4.13), we have

$$\begin{aligned} \| \vec{u} \|_{\mathbb{X}}^2 &\leq \frac{C_{max}}{\beta C_{min}} \| f \|_{L^2} \| \vec{u} \|_{\mathbb{X}}, \\ \| \vec{u} \|_{\mathbb{X}} &\leq \frac{C_{max}}{\beta C_{min}} \| f \|_{L^2}, \end{aligned} \tag{5.4.15}$$

and then,

$$\| p \|_{\mathbb{Q}} \leq \frac{C_{max}^2}{\beta^2 C_{min}} \| f \|_{L^2} . \tag{5.4.16}$$

Now it is easy task to get the bound (5.4.11).

5.4.3 Karhunen. Lo‘eve (KL) expansion

We use the well known KL expansion to express $K^{-1}(x,\omega)$ as a summation of scaled product of two functions one of these function is deterministic while the second is random variable. This expression is useful to transform the saddle point problem (5.4.10) into one which can be solved by deterministic numerical methods. There are several expansions are available (see [57] for a survey). In this dissertation, we focus on the Karhunen. Lo‘eve (KL) expansion (see [87], [65], [46] and [66] for the (KL) expansion)

$$K^{-1}(x,\omega) = k_0(x) + \sum_{m=1}^{\infty} \sqrt{\lambda_m} k_m(x) \xi_m(\omega), \tag{5.4.17}$$

In the above expression (5.4.17), $k_0(x)$ denotes the mean of $K^{-1}(x,\omega)$ and $\{(\lambda_m, k_m)\}_{m=1}^{\infty}$ are the eigenpairs of the integral operator $C : L^2(D) \to L^2(D)$, defined by

$$(Cu)(x) = \int_D u(y)c(x,y)dy, \tag{5.4.18}$$

where c is the covariance function of K^{-1} (some popular choices of c are mentioned in [33]. The linear operator C defined by (5.4.18) is compact, self adjoint, and positive. Therefore, the eigenfunctions of C form an orthonormal basis of $L^2(D)$ and the eigenvalues are all positive real numbers with only one accumulation point, namely 0 (see Theorem B.2.1 in [60]). The random variables $\{\xi_m\}_{m=1}^{\infty}$ are uncorrelated random variables in $L^2_{\mathbb{P}}(\Omega)$ with $\mathbf{E}[\xi_m] = 0$ and $\mathbf{E}[\xi_m^2] = 1$ and they are determined by

$$\xi_m(\omega) = \frac{1}{\sqrt{\lambda_m}} \int_D (K^{-1}(x,\omega) - k_0(x))k_m(x)dx. \tag{5.4.19}$$

5.4.4 The weak formulation of the perturbed problem

To discretize the saddle point problem (5.4.10), we follow ([8], [9], [40], [67], [66], [87], [13] and [33]) by truncating (5.4.20) after M terms

$$K_M^{-1}(x,\omega) \approx k_0(x) + \sum_{m=1}^{M} \sqrt{\lambda_m} k_m(x)\xi_m(\omega), \tag{5.4.20}$$

where M here is called the order of the KL decomposition. The convergence of the error $\| K^{-1} - K_M^{-1} \|_{L^\infty(D\times\Omega)}$ to zero depends on how do the eigenvalues λ_m decreasing to zero. For more detail on this convergence we refer to see ([40]) for instance. Now, we put K_M^{-1} instead of K^{-1} in (5.4.10) to get the perturbed problem: find $\vec{u}^M \in \mathbb{X}$

and $p^M \in \mathbb{Q}$

$$\begin{aligned} a_M(\vec{u}^M, \vec{v}) + b(\vec{v}, p^M) &= 0 \qquad\qquad \vec{v}, \in \mathbb{X}, \\ b(\vec{u}^M, w) &= -\langle (f, w)\rangle \qquad w \in \mathbb{Q}, \end{aligned} \tag{5.4.21}$$

where,

$$a_M(\vec{u}, \vec{v}) = \mathbf{E}[\int_D K_M^{-1}\vec{u}\cdot\vec{v}dx] \ \ \forall \vec{u}, \vec{v} \in \mathbb{X} \tag{5.4.22}$$

The well-posedness of the solution to (5.4.21) can be derived under the following assumption

Assumption 3

$$0 < K_{min} \leq K_M^{-1}(x,\omega) \leq K_{max} < \infty \ a.e.\ in\ D \times \Omega, \tag{5.4.23}$$

where K_{max} and K_{min} are positive constants (depending on M, C_{min} and C_{max}).

Theorem 6 *Let $(\vec{u}, p) \in \mathbb{X}\times\mathbb{Q}$ be the solution to (5.4.10) and let $(\vec{u}^M, p^M) \in \mathbb{X}\times\mathbb{Q}$ be the solution to (5.4.21) then*

$$\begin{aligned} \| \vec{u} - \vec{u}^M \|_{\mathbb{X}} &\leq \frac{C_{max}}{\beta C_{min} K_{min}} \| K^{-1} - K_M^{-1} \|_{L^\infty(D\times\Omega)} \| f \|_{L^2(D)}, \\ \| p - p^M \|_{\mathbb{Q}} &\leq \frac{C_{max}}{\beta^2 C_{min}}(1 + \frac{K_{max}}{K_{min}}) \| K^{-1} - K_M^{-1} \|_{L^\infty(D\times\Omega)} \| f \|_{L^2(D)}. \end{aligned} \tag{5.4.24}$$

Proof: Let $e_u = \vec{u} - \vec{u}^M \in \mathbb{X}$, $e_p = p - p^M \in \mathbb{Q}$. Then from (5.4.10) and (5.4.21), we have

$$\begin{aligned} a(\vec{u}, e_u) &= -b(e_u, p) = 0, \\ a_M(\vec{u}^M, e_u) &= -b(e_u, p^M) = 0 \end{aligned} \tag{5.4.25}$$

and, hence

$$K_{min} \| e_u \|_{\mathbb{X}}^2 \leq a_M(e_u, e_u) = a(\vec{u}, e_u) - a_M(\vec{u}, e_u). \tag{5.4.26}$$

Using the above equations and the bound given in (5.4.15), we have

$$\begin{aligned}
K_{min} \parallel e_u \parallel_{\mathbb{X}}^2 &\leq a(\vec{u}, e_u) - a_M(\vec{u}, e_u) \\
K_{min} \parallel e_u \parallel_{\mathbb{X}}^2 &\leq \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel \vec{u} \parallel_{\mathbb{X}} \parallel e_u \parallel_{\mathbb{X}} \\
\parallel e_u \parallel_{\mathbb{X}} &\leq \frac{1}{K_{min}} \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel \vec{u} \parallel_{\mathbb{X}} \\
\parallel \vec{u} - \vec{u}^M \parallel_{\mathbb{X}} &\leq \frac{C_{max}}{\beta C_{min} K_{min}} \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel f \parallel_{L^2} .
\end{aligned} \tag{5.4.27}$$

Similarly for e_p. From the inf-sup condition we have

$$\underset{\vec{r}\in\mathbb{X}}{sup} \frac{| b(\vec{r}, e_p) |}{\parallel \vec{r} \parallel_{\mathbb{X}}} \geq \beta \parallel e_p \parallel_{\mathbb{Q}} . \tag{5.4.28}$$

So, we have

$$\begin{aligned}
b(\vec{r}, e_p) &= b(\vec{r}, p) - b(\vec{r}, p^M) \\
&= -a(\vec{u}, \vec{r}) + a_M(u^M, \vec{r}) \\
&= -a(\vec{u}, \vec{r}) + a_M(\vec{u}, \vec{r}) - a_M(e_u, \vec{r}).
\end{aligned} \tag{5.4.29}$$

$$| b(\vec{r}, e_p) | \leq \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel \vec{u} \parallel_{\mathbb{X}} \parallel \vec{r} \parallel_{\mathbb{X}} + K_{max} \parallel e_u \parallel_{\mathbb{X}} \parallel \vec{r} \parallel_{\mathbb{X}} \tag{5.4.30}$$

$$| b(\vec{r}, e_p) | \leq (K_{max} \parallel e_u \parallel_{\mathbb{X}} + \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel \vec{u} \parallel_{\mathbb{X}}) \parallel \vec{r} \parallel_{\mathbb{X}} . \tag{5.4.31}$$

Now, use the bound (5.4.15) to have

$$\begin{aligned}
\parallel e_p \parallel_{\mathbb{Q}} &\leq \frac{1}{\beta}(K_{max} \parallel e_u \parallel_{\mathbb{X}} + \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel \vec{u} \parallel_{\mathbb{X}}) \\
\parallel e_p \parallel_{\mathbb{Q}} &\leq (\frac{K_{max} C_{max}}{\beta^2 K_{min} C_{min}} + \frac{C_{max}}{\beta^2 C_{min}}) \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel f \parallel_{L^2} \\
\parallel p - p^M \parallel_{\mathbb{Q}} &\leq \frac{C_{max}}{\beta^2 C_{min}}(1 + \frac{K_{max}}{K_{min}}) \parallel K^{-1} - K_M^{-1} \parallel_{L^\infty(D\times\Omega)} \parallel f \parallel_{L^2} .
\end{aligned} \tag{5.4.32}$$

5.5 Deterministic spaces approximation

Let T_h be a triangulations of D, h denotes the maximal diameter of the elements of T_h. Now for a given h and T_h, the space X_h^k of discrete velocities that approximates the space X is defined by $X_h^k = \{\vec{v}_h \in X;\ \ \forall K \in T_h, v_h \mid_K \in P_k(K)^2\}$. The space Q_h^k of the discrete pressure that approximates Q is defined by $Q_h^k = \{w_h \in Q;\ \ \forall K \in T_h, w_h \mid_K \in P_{k+1}(K)\}$. Here $P_k(K)$ denotes the space of polynomials of degree $\leq k$ on triangle K. Note that any function $\vec{v}_h$ in X_h has a zero divergence on each element K and hence on D. For $k = 0$, the discrete spaces have the special piecewise forms,

$$\begin{aligned} X_h = X_h^0 = \{\vec{v}_h \in X : \vec{v}_h \mid_K \in P_0(K)^2\}, \\ Q_h = Q_h^0 = \{w_h \in Q : w_h \mid_K \in P_1(K)\}. \end{aligned} \tag{5.5.1}$$

The functions $(\chi_K, 0)$ and $(0, \chi_K)$ for all $K \in T_h$ form an orthonormal basis for X_h where χ_K is the characteristic function of the triangle K. In analogy with the continuous case, we introduce the discrete kernel $Z_h = \{\vec{v}_h \in X_h : b(\vec{v}_h, w_h) = 0\ \ \forall w_h \in Q_h\}$. Now we look for $\vec{u}_h \in X_h$ and $p_h \in Q_h$ such that

$$\begin{aligned} \mathbf{a}(\vec{u}_h, \vec{v}_h) + \mathbf{b}(\vec{v}_h, p_h) = 0 \qquad & \vec{v}_h \in X_h, \\ \mathbf{b}(\vec{u}_h, w_h) = \mathbf{l}(w) \qquad & w_h \in Q_h. \end{aligned} \tag{5.5.2}$$

The continuity of $a(.,.)$ on $X_h \times X_h$ and $b(.,.)$ on $X_h \times Q_h$, the coercivity of $a(.,.)$ on Z_h can be done as in section 4.1. For the inf-sup condition: for any w_h in Q_h, we take $\vec{r}_h \doteq \nabla w_h$ and obtain the following inf-sup condition

$$\forall\ w_h \in Q_h, \quad \underset{\vec{r}_h \in X_h}{sup} \frac{\mid b(\vec{r}_h, w_h) \mid}{\parallel \vec{r}_h \parallel_{[L^2(D)]^2}} \geq \tilde{\beta} \parallel w_h \parallel_{H^1(D)}, \tag{5.5.3}$$

where $\tilde{\beta}$ is a positive constant (the discrete inf-sup constant) independent of h. The above argument gives the existence and uniqueness of the solution $(\vec{u}_h, p_h)$ to (5.5.2). Now, let X_h=span $\{\varphi_i\}_{i=1}^n$ and Q_h = span $\{\phi_i\}_{i=1}^m$ where n = twice number of the triangles in the mesh and m = number of nodes. Since $\vec{u}_h \in X_h$ and $p_h \in Q_h$ then

$$\vec{u}_h = \sum_{i=1}^{n} u_i \varphi_i \qquad p_h = \sum_{i=1}^{m} p_i \phi_i, \tag{5.5.4}$$

as well as the basis of test functions $v = \varphi_i$, $i = 1, ...n$ and $w = \phi_j$, $j = 1, ..., m$. So we put these in (5.5.2), to get the saddle point problem

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \underline{u} \\ \underline{p} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{f}, \end{bmatrix}, \tag{5.5.5}$$

where $\underline{u} = [u_1, ..., u_n]^T$, $\underline{p} = [p_1, ..., p_m]^T$. The block matrices A and B in (5.5.5) are defined by $[A]_{i,j} = (K^{-1}\varphi_i, \varphi_j)$ for $1 \le i, j \le n$ and $[B]_{j,i} = -(\varphi_i, \nabla\phi_j)$ for $1 \le i \le n$ and $1 \le j \le m$. The data $\underline{f}$ is given by $[f]_j = -(f, \phi_j)$ for $1 \le j \le m$.

5.6 Finite-dimensional noise

Since the random variables appearing in the KL– expansion are only uncorrelated, we need to the following assumption, (see [33] and [13])

Assumption 4 *The random variables ξ_m in the expansion (5.4.17) are independent.*

Let $\Gamma_m := \xi_m(\omega),\ m = 1, ..., M$ be bounded intervals in $\mathbb{R}$, and assume that the density functions$\rho_m : \Gamma_m \longrightarrow \mathbb{R}^+$ of all ξ_m are given. Let $\Gamma := \Gamma_1 \times ... \times \Gamma_M \subset \mathbb{R}^M$ denotes the range of the M-dimensional random vector $\xi = (\xi_1, ..., \xi_M)$ and let $y = (y_1, ..., y_M)$ be a vector in Γ where $y_j = \xi_j(\omega);\ j = 1, ..., M$. From Assumption 3, the joint density function of ξ can be written as

$$\rho(y) = \rho_1(y_1)...\rho_M(y_M), \tag{5.6.1}$$

where ρ_i is the density function of ξ_i. The Doob Dynkin lemma [75] allows us to write the velocity and the pressure as functions of x and ξ. Now one can turns the original stochastic equations (5.4.10) into deterministic parametric equations as follows

$$\begin{aligned} a_M(\vec{u}^M, \vec{v}) + b(\vec{v}, p^M) &= 0 & \vec{v} \in \mathbf{X}, \\ b(\vec{u}^M, w) &= -\int_\Gamma \rho(y) \int_D f \cdot w dx dy & w \in \mathbf{Q}, \end{aligned} \tag{5.6.2}$$

where $\mathbf{X} := X \otimes L^2_\rho(\Gamma)$ and $\mathbf{Q} := Q \otimes L^2_\rho(\Gamma)$, with norms $\| v \|_{\mathbf{X}} := (\int_\Gamma \| \vec{v} \|^2_X \rho(y) dy)^{\frac{1}{2}}$ and $\| w \|_{\mathbf{Q}} := (\int_\Gamma \| w \|^2_{H^1(D)} \rho(y) dy)^{\frac{1}{2}}$. The first bilinear forms $a_M(.,.)$ in (5.6.2) is defined by

$$a_M(\vec{u}, \vec{v}) = \int_\Gamma \rho(y) \int_D K_M^{-1} \vec{u} \cdot \vec{v} dx dy \quad \forall \vec{u}, \vec{v} \in \mathbf{X}. \tag{5.6.3}$$

where $K_M^{-1}(.,.)$ contains the parameterized coefficient

$$K_M^{-1}(x, y) \approx k_0(x) + \sum_{m=1}^{M} \sqrt{\lambda_m} k_m(x) y_m. \tag{5.6.4}$$

The well posedness of the solution pair $(\vec{u}(x, y), p(x, y)) \in \mathbf{X} \times \mathbf{Q}$ can be achieved in analogs with problem (5.4.10) with assuming that $K_M^{-1}(x, y)$ is bounded as in Assumption 3 and replacing the tensor product spaces $\mathbb{X}$ and $\mathbb{Q}$ by the spaces $\mathbf{X}$ and $\mathbf{Q}$, respectively and $(\Omega, \mathbb{F}, \mathbb{P})$ by $(\Gamma, \mathbb{B}, \rho dy)$ where $\rho : \Gamma \to \mathbb{R}^+$ is the joint probability density function of the vector y and $\mathbb{B}$ is the Borel sigma algebra generated by Γ.

5.7 Stochastic spaces approximation

As in the usual way, we need to chose suitable finite-dimensional subspaces $X_{h,p} \subset \mathbf{X}$ and $Q_{h,p} \subset \mathbf{Q}$. These subspaces can be constructed by finding finite subspaces for their components spaces X, Q, and $L^2_\rho(\Gamma)$. In section 5, we introduced finite dimensional subspaces for the spaces X and Q and now we have to construct a subspace $\Psi_p(\Gamma) \subset L^2_\rho(\Gamma)$ of demential $N_\xi < \infty$.

There are several constructions for Ψ_p in the literature. Tow of them are the tensor product polynomials (TP) spaces and the complete polynomial (CP) spaces (see [101, 68, 42], [44], [66], [43], [58] for using the complete polynomial (CP) spaces and see also [8], [13], [40], [28], [56], [82] for using the tensor product polynomial (TP) spaces).

It is shown that when using a TP space as a basis, there is a basis, (called double orthogonal), can be constructed for which the stochastic matrices are block diagonal see [8], [9]. This diagonally properties of the stochastic matrices leads to decouple the large system into small systems.

Using (TP) polynomials, the result is that $dim(\Psi_p) = N_\xi = (p+1)^M$ where M is the order of the (KL) expansion.

The best advantage of using (TP) polynomials (which are discussed in [8], [9], [28] and [56]) is that one can decouple the resulted system into N_ξ systems each of which is of dimension $N_x = N_u + N_q$.

On the other hand,, if the complete polynomials (CP) space is used. In this case, we obtain a basis of $dim(\Psi_p) = N_\xi = \frac{(M+P)!}{M!P!}$. As discussed in [34], there is no basis as in (TP) space in which the coupled linear system can be decoupled into smaller systems, and therefore, a coupled system os size $N_x N_\xi$ must be solved.

In [35], it is shown that when all the random variables in the (KL) expansion are uniformly distributed, a basis of Legendre polynomials is used [34].
Let $\Psi_p(\Gamma) =$ span $\{\Psi_i : i = 1, ..., N_\xi\}$, the discrete tensor product spaces are given by $\mathbf{X}_{h,p} = X_h \otimes \Psi_p$ and $\mathbf{Q}_{h,p} = Q_h \otimes \Psi_p$. We introduce the tensor discrete kernel $Z_{hp} = \{\vec{v}_{h,p} \in X_{h,p} : b(\vec{v}_{h,p}, w_{h,p}) = 0 \quad \forall w_{h,p} \in Q_{h,p}\}$. Then the discrete version of problem (5.4.21) is to find $\vec{u}^M_{h,p} \in \mathbf{X}_{h,p}$ and $p^M_{h,p} \in \mathbf{Q}_{h,p}$ such that

$$\begin{aligned} a_M(\vec{u}^M_{h,p}, \vec{v}_{h,p}) + b(\vec{v}_{h,p}, p^M_{h,p}) &= 0; & \vec{v}_{h,p} \in \mathbf{X}_{h,p}, \\ b(\vec{u}^M_{h,p}, w_{h,p}) &= -\int_\Gamma \rho(y) \int_D f \cdot w_{h,p} dx dy; & w_{h,p} \in \mathbf{Q}_{h,p}. \end{aligned} \tag{5.7.1}$$

The continuity of $a(.,.)$ on $\mathbf{X}_{h,p} \times \mathbf{X}_{h,p}$ and $b(.,.)$ on $\mathbf{X}_{h,p} \times \mathbf{Q}_{h,p}$, the coercivity of $a(.,.)$ on $Z_{h,p}$ can be showed as in section 4.1. For the inf-sup condition: for any $w_{h,p}$ in $\mathbf{Q}_{h,p}$, we take $\vec{r}_{h,p}$ equals to $\nabla w_{h,p}$ and obtain the following inf-sup condition

$$\forall\ w \in \mathbf{Q}_{h,p} \quad \sup_{\vec{r} \in \mathbf{X}_{h,p}} \frac{| \ b(\vec{r}, w) \ |}{\| \ \vec{r} \ \|_{[L^2]^2 \otimes L^2_\rho(\Gamma)}} \geq \tilde{\beta} \ \| \ w \ \|_{H^1 \otimes L^2_\rho(\Gamma)}, \tag{5.7.2}$$

where $\tilde{\beta} > 0$ is the discrete inf-sup constant for (5.5.3) and is independent of h and p. The above discussion gives the existence and uniqueness of the solution to (5.7.1).

Theorem 7 *Let $h > 0$ and let $p \geq 1$, then the discrete problem (5.7.1) has a unique solution and the following bounds for the truncation error holds*

$$\begin{aligned} &\| \ \vec{u}^M - \vec{u}^M_{h,p} \ \|_{\boldsymbol{X}} \leq (1 + \frac{C}{\tilde{\beta}}) \inf_{\vec{v} \in \boldsymbol{X}_{h,p}} \ \| \ \vec{u}^M - \vec{v} \ \|_{\boldsymbol{X}}, \\ &\| \ p^M - p^M_{h,p} \ \|_{\boldsymbol{Q}} \leq \frac{K_{max}}{\tilde{\beta}} \ \| \ \vec{u}^M - \vec{u}^M_{h,p} \ \|_{\boldsymbol{X}} + (1 + \frac{1}{\tilde{\beta}}) \inf_{q \in \boldsymbol{Q}_{h,p}} \ \| \ q - p^M \ \|_{\boldsymbol{Q}}, \end{aligned} \tag{5.7.3}$$

where $(\vec{u}^M, p^M) \in \mathbb{X} \times \mathbb{Q}$ *is the solution of (5.6.2) and* $(\vec{u}^M_{h,p}, p^M_{h,p}) \in \boldsymbol{X}_{h,p} \times \boldsymbol{Q}_{h,p}$ *is the solution of (5.7.1). The constants* K_{min}, K_{max} *and* $\tilde{\beta}$ *are defined above (analog of this theorem is given in Lemma 3.1. in [33]).*

proof: Let $e^h_u = \vec{u}^M - \vec{u}^M_{h,p} \quad \in \mathbf{X}_h$ and $e^h_p = p^M - p^M_{h,p} \quad \in \mathbf{Q}_h$.

The orthogonality here is $b(e^h_u, w) = 0,\ \forall w \in \mathbf{Q}_h$. For any $\vec{v} \in \mathbf{X}_h$, we have

$$
\begin{aligned}
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} = \| \vec{u}^M - \vec{v} + \vec{v} - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} + \| \vec{v} - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} + \frac{1}{\tilde{\beta}} \sup_{w \in \mathbf{Q}_h} \frac{| b(\vec{v} - \vec{u}^M_{h,p}, w) |}{\| w \|_{\mathbf{Q}_h}} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} + \frac{1}{\tilde{\beta}} \sup_{w \in \mathbf{Q}_h} \frac{| b(\vec{v} - \vec{u}^M + \vec{u}^M - \vec{u}^M_{h,p}, w) |}{\| w \|_{\mathbf{Q}_h}} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} + \frac{1}{\tilde{\beta}} \sup_{w \in \mathbf{Q}_h} \frac{| b(\vec{v} - \vec{u}^M, w) | + | b(\vec{u}^M - \vec{u}^M_{h,p}, w) |}{\| w \|_{\mathbf{Q}_h}} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} + \frac{1}{\tilde{\beta}} \sup_{w \in \mathbf{Q}_h} \frac{| b(\vec{v} - \vec{u}^M, w) |}{\| w \|_{\mathbf{Q}_h}} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} + \frac{C}{\tilde{\beta}} \| u^M - v \|_{\mathbf{X}_h} \\
&\| \vec{u}^M - \vec{u}^M_{h,p} \|_{\mathbf{X}_h} \leq (1 + \frac{C}{\tilde{\beta}}) \inf_{\vec{v} \in \mathbf{X}_h} \| \vec{u}^M - \vec{v} \|_{\mathbf{X}_h} .
\end{aligned}
\tag{5.7.4}
$$

Similarly for the discrete error of the presser, for any $q \in \mathbf{Q}_h$

$$
\begin{aligned}
\tilde{\beta} \| q - p^M_{h,p} \|_{\mathbf{Q}_h} &\leq \sup_{\vec{v} \in \mathbf{X}_h} \frac{| b(\vec{v}, q - p^M_{h,p}) |}{\| v \|_{\mathbf{X}_h}} \\
&= \sup_{\vec{v} \in \mathbf{X}_h} \frac{| b(\vec{v}, p^M - p^M_{h,p}) + b(\vec{v}, q - p^M) |}{\| \vec{v} \|_{\mathbf{X}_h}} \\
&= \sup_{\vec{v} \in \mathbf{X}_h} \frac{| -a(u^M - u^M_{h,p}, \vec{v}) + b(\vec{v}, q - p^M) |}{\| \vec{v} \|_{\mathbf{X}_h}}.
\end{aligned}
\tag{5.7.5}
$$

So, we have

$$\begin{aligned}\tilde{\beta} \parallel q - p^M_{h,p} \parallel_{\mathbf{Q}_h} &\leq K_{max}(\parallel \vec{u}^M - \vec{u}^M_{h,p} \parallel) + C \parallel q - p^M \parallel \\ \parallel q - p^M_{h,p} \parallel_{\mathbf{Q}_h} &\leq \frac{K_{max}}{\tilde{\beta}}(\parallel \vec{u}^M - \vec{u}^M_{h,p} \parallel) + \frac{C}{\tilde{\beta}} \parallel q - p^M \parallel,\end{aligned} \tag{5.7.6}$$

add $\parallel p^M - q \parallel$ to both sides of the last equation

$$\begin{aligned}\parallel p^M - p^M_{h,p} \parallel_{\mathbf{Q}_h} &\leq \frac{K_{max}}{\tilde{\beta}}(\parallel \vec{u}^M - \vec{u}^M_{h,p} \parallel) + \frac{C}{\tilde{\beta}} \parallel q - p^M \parallel + \parallel p^M - q \parallel \\ \parallel p^M - p^M_{h,p} \parallel_{\mathbf{Q}_h} &\leq \frac{K_{max}}{\tilde{\beta}}(\parallel \vec{u}^M - \vec{u}^M_{h,p} \parallel) + (1 + \frac{C}{\tilde{\beta}}) \parallel q - p^M \parallel \\ \parallel p^M - p^M_{h,p} \parallel_{\mathbf{Q}_h} &\leq (\frac{K_{max}}{\tilde{\beta}} + \frac{K^2_{max}}{\tilde{\beta}^2}) \underset{\vec{v} \in \mathbf{X}_h}{inf} (\parallel \vec{u}^M - \vec{v} \parallel) + (1 + \frac{C}{\tilde{\beta}}) \underset{q \in \mathbf{Q}_h}{inf} \parallel q - p^M \parallel .\end{aligned} \tag{5.7.7}$$

Since, we have

$$\begin{aligned}\vec{u} - \vec{u}^M_{h,p} &= \vec{u} - \vec{u}^M + \vec{u}^M - \vec{u}^M_{h,p} \\ p - p^M_{h,p} &= p - p^M + p^M - p^M_{h,p}.\end{aligned} \tag{5.7.8}$$

Then, one can easily find the full error $\parallel u - u^M_{h,p} \parallel_{\mathbf{X}_h}$ and $\parallel p - p^M_{h,p} \parallel_{\mathbf{Q}_h}$.

5.8 Stochastic matrix structures

Inserting representation (5.4.20) of K^{-1} and writing the trail functions as

$$\vec{u}^M_{h,p}(x,\xi) = \sum_{k=1}^{N_\xi} \sum_{i=1}^{N_u} \vec{u}_{i,k} \varphi_i(x) \Psi_k(\xi), \quad p^M_{h,p}(x,\xi) = \sum_{k=1}^{N_\xi} \sum_{i=1}^{N_p} p_{i,k} \phi_i(x) \Psi_k(\xi), \tag{5.8.1}$$

also the test functions as $v(x,\xi) = \varphi_j(x)\Psi_l(\xi)$, $j = 1,..,N_u$; $l = 1,...,N_\xi$ and $w(x,\xi) = \phi_j(x)\Psi_l(\xi)$, $j = 1,..,N_p$; $l = 1,...,N_\xi$ into (5.7.1) to get the saddle-point problem

$$\begin{bmatrix} \hat{A}_1 & \hat{B}_1^T \\ \hat{B}_1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f} \end{bmatrix}. \tag{5.8.2}$$

In the above equation, the vectors **u** and **p** are represented as

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N_\xi} \end{bmatrix}, \qquad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{N_\xi} \end{bmatrix}, \tag{5.8.3}$$

where $u_1, ...u_{N_\xi}$ are vectors of length N_u and $p_1, ...p_{N_\xi}$ are vectors of length N_p. The i_{th} component of u_k is u_{ik} and i_{th} component of p_k is p_{ik}. A similar representation holds for the vector **f**

$$[\mathbf{f}_l]_j = -\langle \Psi_l \rangle (f, \phi_j), \quad j = 1, ..., N_p. \tag{5.8.4}$$

The block matrices $\hat{A}_1 \in \mathbb{R}^{N_u N_\xi \times N_u N_\xi}$ and $\hat{B}_1 \in \mathbb{R}^{N_p N_\xi \times N_p N_\xi}$ in (5.8.2) are given by

$$\begin{aligned} \hat{A}_1 &= \langle (K^{-1}\varphi_i, \varphi_j) \Psi_k \Psi_l \rangle \\ &= \langle (k_0 \varphi_i, \varphi_j) \Psi_k \Psi_l \rangle + \sum_{m=1}^{M} \sqrt{\lambda_m} (k_m \varphi_i, \varphi_j) \langle \xi_m \Psi_k \Psi_l \rangle, \end{aligned} \tag{5.8.5}$$

where $j, i = 1, ..., N_u, \quad l, k = 1, ..., N_\xi$ and

$$\hat{B}_1 = -\langle (\varphi_j, \nabla \phi_i) \Psi_l \Psi_l \rangle = -(\varphi_j, \nabla \phi_i) \langle \xi_m \Psi_k \Psi_l \rangle, \tag{5.8.6}$$

where $j = 1, ..., N_u, \ i = 1, ..., N_p, \ \ l, k = 1, ..., N_\xi$. Integrals with respect to ξ and x can be separated into a product of two integrals. This separation property and the (KL) expansion (5.4.20) implies that the above matrices can be expressed as sums of

Kronecker products as:

$$\hat{A}_1 = G_0 \otimes A_0 + \sum_{m=1}^{M} G_m \otimes A_m, \qquad \hat{B}_1 = G_0 \otimes B, \tag{5.8.7}$$

where

$$[A_0]_{j,i} = (k_0 \varphi_i, \varphi_j) \in \mathbb{R}^{N_u \times N_u}, \quad j, i = 1, ..., N_u, \tag{5.8.8}$$

$$[A_m]_{j,i} = \sqrt{\lambda_m}(k_m \varphi_i, \varphi_j) \in \mathbb{R}^{N_u \times N_u}, \quad j, i = 1, ..., N_u, \tag{5.8.9}$$

$$[B]_{j,i} = -(\varphi_j, \nabla \phi_i) \in \mathbb{R}^{N_u \times N_p}, \quad j, ..., N_u \quad i = 1, ..., N_p, \tag{5.8.10}$$

$$[G_0]_{l,k} = \langle \Psi_k \Psi_l \rangle \in \mathbb{R}^{N_\xi \times N_\xi}, \quad l, k = 1, ..., N_\xi, \tag{5.8.11}$$

$$[G_m]_{l,k} = \langle \xi_m \Psi_k \Psi_l \rangle \in \mathbb{R}^{N_\xi \times N_\xi}, \quad l, k = 1, ..., N_\xi. \tag{5.8.12}$$

5.8.1 Remarks

(i) The matrices A_0 can be seen as the (1,1)-blocks of (5.5.5) with the scaling k_0. Also A_m can be seen as the (1,1)-blocks of (5.5.5) with the permeability scaling $\sqrt{\lambda_m} k_m$. The matrix B is exactly the (2,1)-block of the (5.5.5) with out any extra parameters.

(ii) When we use a basis (TP) mentioned in section 7, the saddle point problem (5.8.2) leads to a coupled system of linear equations, whose dimension is $N_\xi(N_u + N_p)$ which is not easy to solve. In [8], Babuska, Tempone, and Zouraris proposed a particular choice of basis functions (named double orthogonal polynomials) for Ψ_p and they showed that the construction of this basis leads to solve an eigenvalue problem. This basis allows us to decouple the system (5.8.2) into N_ξ saddle point problems each of which of size $N_u + N_p$. In [7], Babuška, Nobile, and Tempone provided a useful characteristic that is, the set $\{\psi_j\}_{j=1}^{p+1}$ of double orthogonal polynomials of degree p satisfy

$$\begin{aligned} &\int_\Gamma \psi_i(y)\psi_j(y)\rho(y)dy = \delta_{ij} \\ &\int_\Gamma y\psi_i(y)\psi_j(y)\rho(y)dy = C_i\delta_{ij}, \end{aligned} \tag{5.8.13}$$

for each $1 \leq i, j \leq p+1$, C_i are the $p+1$ roots of the orthogonal polynomial $w \in \Psi_{p+1}(\Gamma)$ with respect to the weight function ρ, ($\rho : \Gamma \longrightarrow \mathbb{R}$ and $\Gamma \subset \mathbb{R}$), and δ_{ij} is the Kronecker symbol. The above useful characteristic can be seen in details in (Lemma 2.1 in [7]).

(iii) The beauty of using our new formulation is that the matrices A_0 and A_m are diagonal. This good feature comes form the best choice of the discrete spatial spaces. In additional, when we use the doubly orthogonal polynomials to span the tensor product polynomials space that approximates the stochastic space, the matrices G_0 and G_m are diagonal matrices. These diagonal structures make the matrices $\hat{A}_1$ and $\hat{B}_1$ have diagonal structures. Hence, the inversion of the matrix $\hat{A}_1$ becomes an easy task.

5.9 Numerical examples

In this section, we will provide two numerical examples one for the deterministic Darcy's equation (5.4.1) and the second for the stochastic Darcy's equation given in (5.1.5). For the second example, we start by calculating eigenvalues and the corresponding eigenfunctions for the operator given in (5.4.18). All numerical computations were obtained using MATLAB 7 installed on HP-laptop with intel Core 2 Duo CPU processer and with RAM of 4 GB.

5.9.1 Eigenvalue problem

The KL expansion (5.4.20) of K^{-1} requires calculating the eigenpairs of the integral operator given in (5.4.18). To this end, we chose the domain $D = [0,1] \times [0,1]$ and we select the covariance function given in (5.4.18) with $\tau = 2$ and $\sigma = 0.3$ and we approximate the integral by using quadrature approximation formula

$$\int_D u(y)c(x,y)dy \approx \sum_{k=1}^{q} w_k u(p_k) c(x,p_k), \tag{5.9.1}$$

where $\{p_1, ..., p_q\}$ and $w_1, ..., w_q$ are the quadrature and weight points, respectively. From (5.4.18) and (5.9.1) we have

$$\sum_{k=1}^{q} w_k u(p_k) c(x,p_k) = \lambda u(x). \tag{5.9.2}$$

We evaluate (5.9.2) at all points $p_1, ..., p_q$

$$\sum_{k=1}^{q} w_k u(p_k) c(p_i,p_k) = \lambda u(p_i) \quad i = 1, ..., q. \tag{5.9.3}$$

Now (5.9.3) is an eigenvalue problem

$$CU = \lambda U, \tag{5.9.4}$$

$$U = \begin{bmatrix} u(p_1) \\ u(p_2) \\ \vdots \\ u(p_q) \end{bmatrix},$$

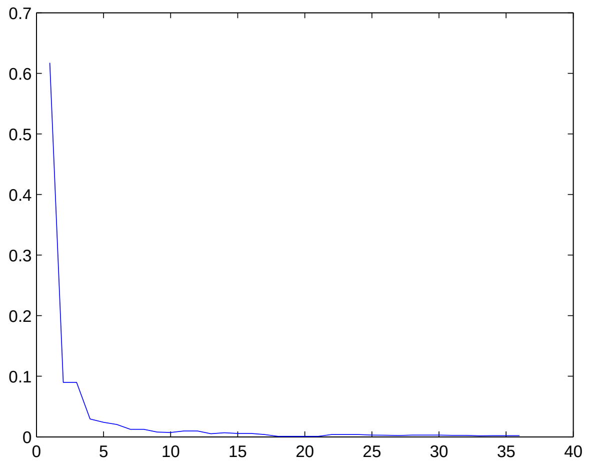


Figure 5 .1: The Eigenvalues

$$C = \begin{bmatrix} w_1 c(p_1,p_1) & w_2 c(p_1,p_2) & \dots & w_q c(p_1,p_q) \\ \vdots & \dots & \dots & \vdots \\ w_1 c(p_q,p_1) & w_2 c(p_q,p_2) & \dots & w_q c(p_q,p_q) \end{bmatrix}.$$

Now it is easy to compute the eigenvalues and the corresponding eigenvectors of the matrix C. Let $\lambda_1, \lambda_2, ..., \lambda_q$ are the eigenvalues and $\bar{u_1}, \bar{u_2}, ..., \bar{u_q}$ are the corresponding eigenvectors of C such that $\bar{u_{ki}} \approx u(p_i)$ where $\bar{u}_k$ is the eigenfunction of (5.4.18) associate with λ_k. We order the eigenvalues from the largest to the smallest and plot them in Figure (5.1). In this figure one can see that the eigenvalues are positive and decreasing to zero. Figure (5.2),..., Figure (5.5) show the first four eigenfunctions corresponding to the first four eigenvalues.

5.9.2 Five-Spot problem(deterministic)

In this subsection, we present numerical results for the problem (5.4.1). We use the test problem (Five-Spot problem) in the domain $D = [0,1] \times [0,1]$ and we put an injection well at the center of D and production wells at the corners of D with no-flow

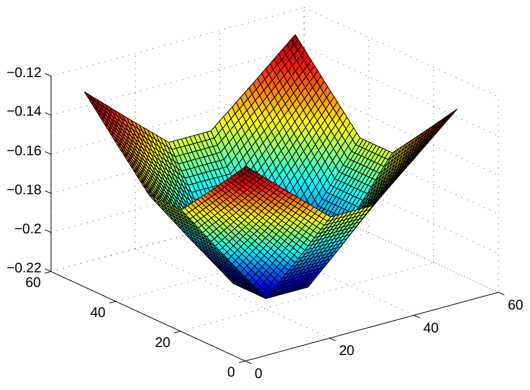


Figure 5 .2: First Eigenfunction

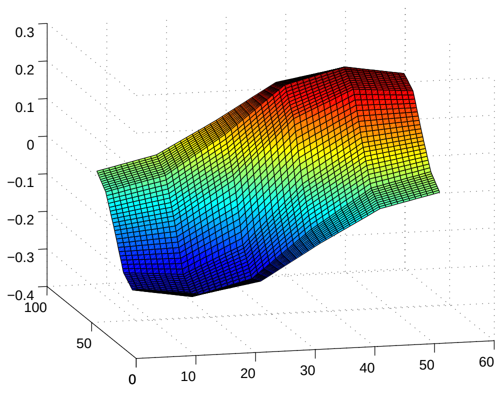


Figure 5 .3: Second Eigenfunction

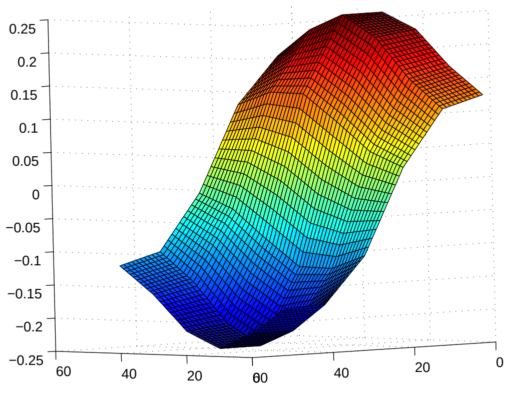


Figure 5 .4: Eigenfunction

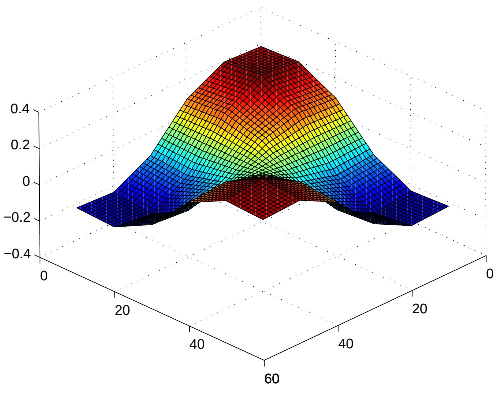


Figure 5 .5: Fourth Eigenfunction

conditions on the boundary. The data $f \equiv 0$ in whole D except at the center and the corners i.e $f(0,0) = f(1,1) = f(1,0) = f(0,1) = -1$ and $f(0.5, 0.5) = 1$. We use the pde-tool box to generate a mesh with 1024 triangles and 545 nodes. Figure (5.6) shows the used discritization mesh. The spaces of the piecewise linear and piecewise constant given in (5.5.1) are used as the discritization spaces for the pressure and the velocity, respectively. Figure (5.7) shows the contour while Figure (5.8) shows the surface of the pressure given in (5.5.5). In these figures the pressure is high at the center (injection well) and low at the corners (production wells) as well as the velocity behavior which is plotted in Figure (5.9). The contour of the velocity components (u^x, u^y) are plotted in Figure (5.10) and Figure (5.11).

5.9.3 Five-Spot problem(stochastic)

In this subsection, we present numerical results for the stochastic problem (5.1.5). We use the same test problem used in the above section (Five-Spot problem) with unknown random permeability K^{-1}. We use the set of tensor product polynomials and we solve N_ξ saddle-point systems each of which is of dimension $(N_q + N_u)$. We use uniform random variables on [-1,1] for the stochastic input and construct the stochastic bases using Legendre polynomials. For the spatial discretization, we generate a mesh with 1024 triangles and 545 nodes. Figure (5.12) shows the used discritization mesh and we use the discrete finite subspaces V_h and W_h as given in Section 5.5 which approximate the velocity and the pressure spaces, respectively. We use the double orthogonal polynomial to spans a basis which approximate the stochastic space. We solved these 81 linear systems by using **the built in MATLAB-linear solver**. Figure (5.13) and Figure (5.14) show the surface and the cantor of the pressure mean respectively. In these Figures, the pressure is high at the center and

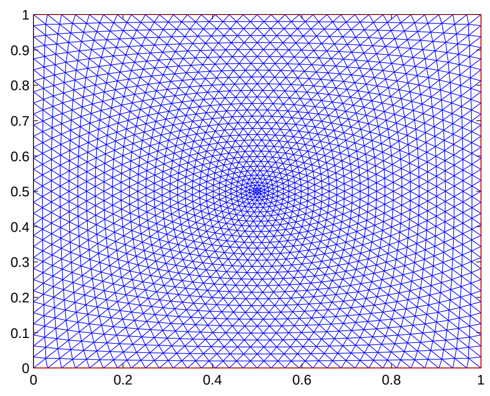


Figure 5 .6: Shape of the Mesh

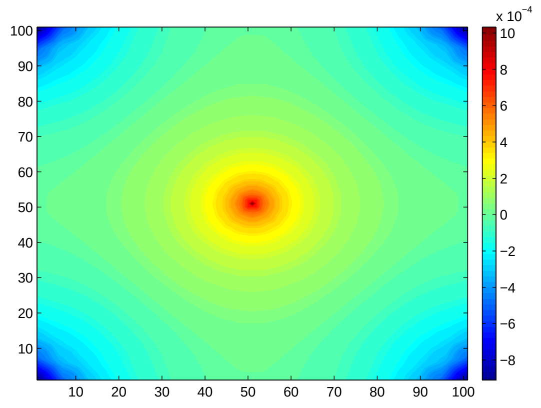


Figure 5 .7: Pressure Contour

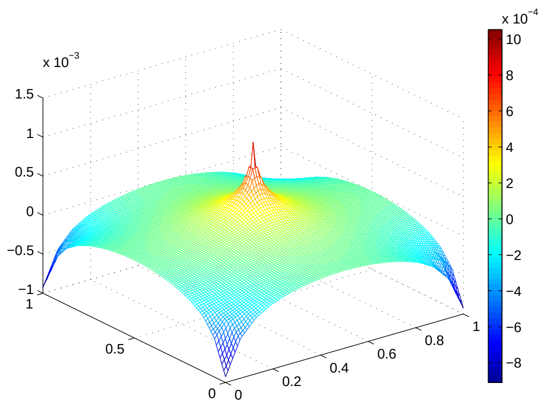


Figure 5 .8: Pressure Surface

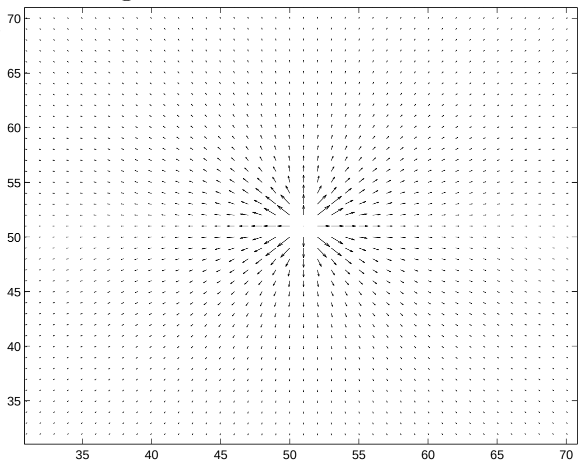


Figure 5 .9: Velocity Distribution

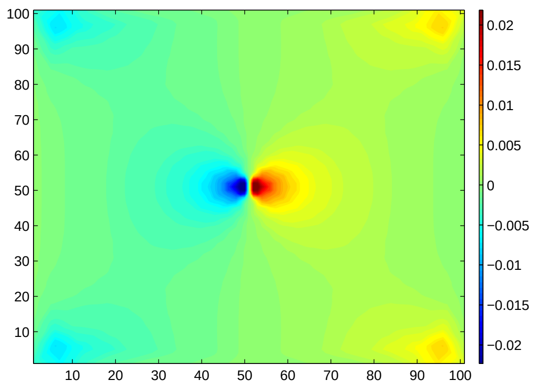


Figure 5 .10: The Velocity of u^x

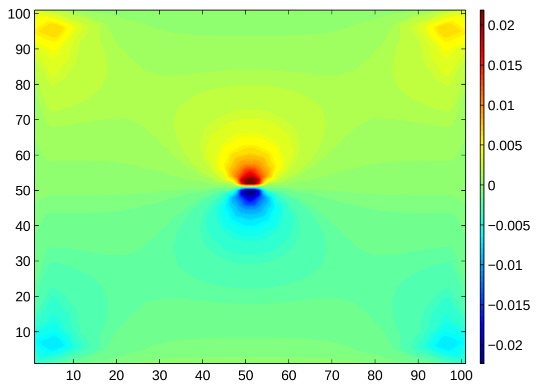


Figure 5 .11: The Velocity of u^y

low at the corners the same as the deterministic problem. The variance of the pressure is plotted in Figure (5.15). The velocity means of the (x, y) components $(\mathbf{u}^x, \mathbf{u}^y)$ are plotted in Figure (5.16) and Figure (5.17) while variances of $(\mathbf{u}^x, \mathbf{u}^y)$ are plotted in Figure (5.18) and Figure (5.19). For more details in the stochastic computation, see the following.

Velocities and pressures mean and variance Calculation

Here, we show the process of computing the mean and variance to the velocities and pressure. Since we have

$$\vec{u}_{h,p}^M(x,\xi) = \sum_{k=1}^{N_\xi}\sum_{i=1}^{N_u} \vec{u}_{i,k}\varphi_i(x)\Psi_k(\xi), \qquad p_{h,p}^M(x,\xi) = \sum_{k=1}^{N_\xi}\sum_{i=1}^{N_p} p_{i,k}\phi_i(x)\Psi_k(\xi), \tag{5.9.5}$$

Once $\mathbf{u}$ and $\mathbf{p}$ have been computed it can be post-processed to obtain the mean and variance of $\vec{u}_{h,p}^M(x,\xi)$ and $p_{h,p}^M(x,\xi)$. We start by the pressure and then by the velocities.

Pressure mean:

$$p_{h,p}^M(x,y) = \sum_{k=1}^{N_\xi}\sum_{i=1}^{N_p} p_{i,k}\phi_i(x)\Psi_k(y) = \sum_{k=1}^{N_\xi}[\sum_{i=1}^{N_p} p_{i,k}\phi_i(x)]\Psi_k(y) = \sum_{k=1}^{N_\xi} P_k\Psi_k(y), \tag{5.9.6}$$

where $P_k = \sum_{i=1}^{N_p} p_{i,k}\phi_i(x)$ is the $k-th$ column of $\mathbf{p}$. Now from the definition of the expectation, we have

$$\begin{aligned}\mathbf{E}[p_{h,p}^M(x,y)] &= \int_\Gamma p_{h,p}^M(x,y)\rho(y)dy = \int_\Gamma \sum_{k=1}^{N_\xi} P_k\Psi_k(y)\rho(y)dy \\ &= \sum_{k=1}^{N_\xi} P_k \int_\Gamma \Psi_k(y)\rho(y)dy = \sum_{k=1}^{N_\xi} P_k\mathbf{E}[\Psi_k(y)].\end{aligned} \tag{5.9.7}$$

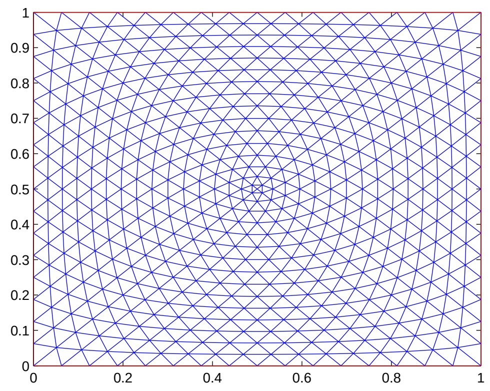


Figure 5 .12: The Mesh

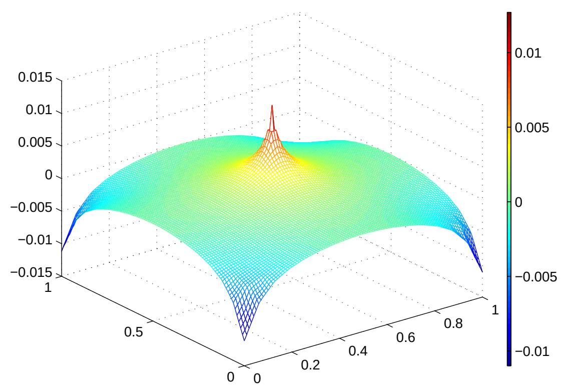


Figure 5 .13: Pressure-mean Surface

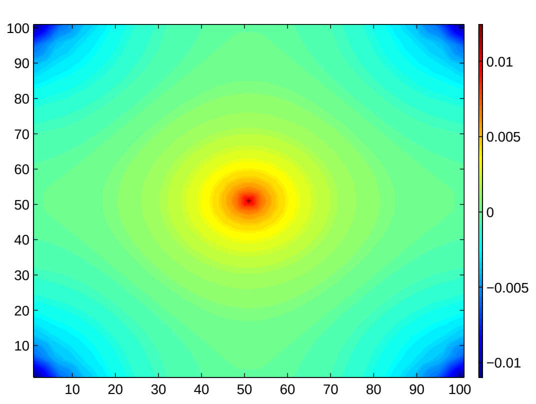


Figure 5 .14: Pressure-mean Contour

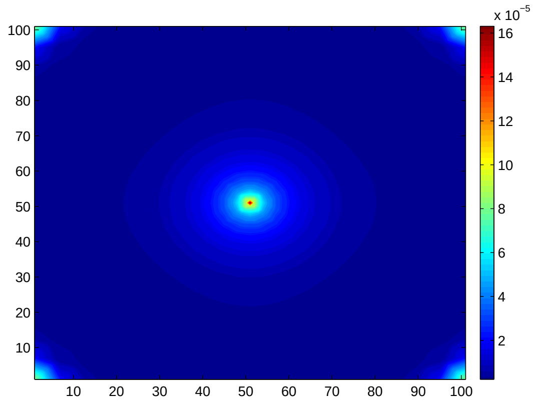


Figure 5 .15: Pressure Variance

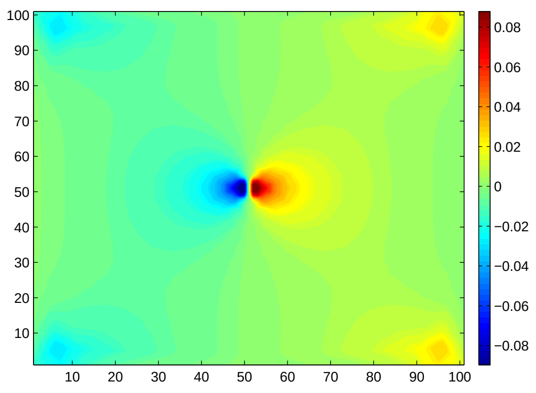


Figure 5 .16: Mean of $\mathbf{u}^x$

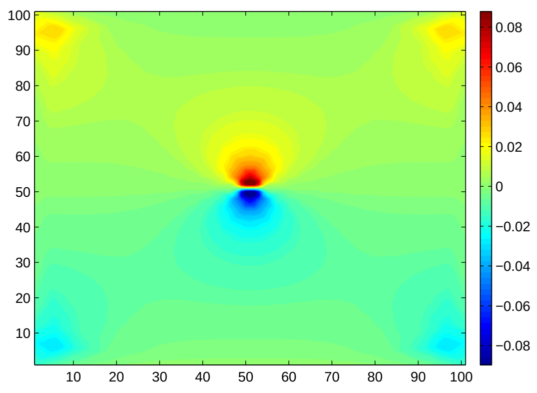
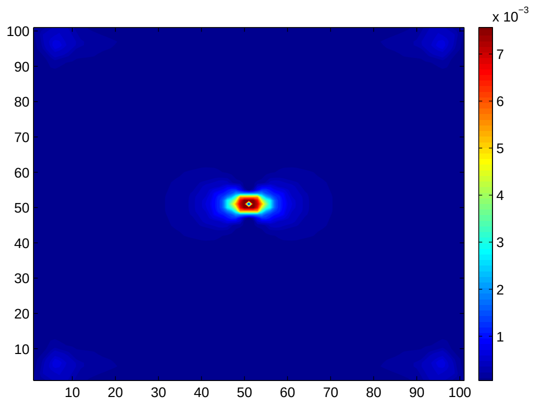
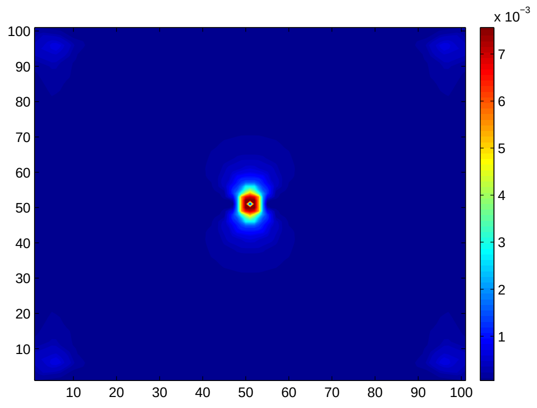


Figure 5 .17: Mean of $\mathbf{u}^y$





Where any element in the set $\{\Psi_k(y)\}_{k=1}^{N_\xi}$ is a product of double orthogonal polynomials $\{\psi_i\}_{i=1}^{p+1}$ where p is the total degree of the Legender polynomial. For example for $M = 4$ and $p = 2$ we have $\Psi_k = \prod_{j=1}^{M=4} \psi_j(y_j)$. In this case

$$\begin{aligned}\Psi_1 &= \psi_1(y_1)\psi_1(y_2)\psi_1(y_3)\psi_1(y_4),\\ \Psi_2 &= \psi_1(y_1)\psi_1(y_2)\psi_1(y_3)\psi_2(y_4),\\ \Psi_3 &= \psi_1(y_1)\psi_1(y_2)\psi_1(y_3)\psi_3(y_4).\end{aligned} \tag{5.9.8}$$

and so on.

Pressure variance:

We know that $\mathbf{Var}[p_{h,p}^M] = \mathbf{E}[{p_{h,p}^M}^2] - (\mathbf{E}[p_{h,p}^M])^2$. The second moment of the pressure is calculated as follow

$$\mathbf{E}[{p_{h,p}^M}^2] = \langle p_{h,p}^M, p_{h,p}^M\rangle = \langle \sum_{k=1}^{N_\xi} P_k\Psi_k(y), \sum_{k=1}^{N_\xi} P_k\Psi_k(y)\rangle = \sum_{k=1}^{N_\xi} P_kP_k\langle \Psi_k(y), \Psi_k(y)\rangle = \sum_{k=1}^{N_\xi} {P_k}^2. \tag{5.9.9}$$

Then $\mathbf{Var}[p_{h,p}^M] = \sum_{k=1}^{N_\xi} {P_k}^2 - (\mathbf{E}[p_{h,p}^M])^2$.

For the velocities, at the beginning, we consider the odd row components to be v^x and the even to be v^y and then we compute both v^x and v^y at the nodes (because we calculated the velocities in each triangles). After this step, we evaluate the v^x–mean, v^y–mean, v^x–variance and v^y– variance as we did for the pressures.

Stochastic matrices

We have

$$[G_m]_{l,k} = \langle \xi_m\Psi_k\Psi_l\rangle, \tag{5.9.10}$$

where Ψ_k is a product of the double orthogonal polynomials ψ_k. Hence, all the matrices G_m are diagonal. This means that

$$[G_m]_{k,k} = \langle \xi_m \Psi_k \Psi_k \rangle = \langle \xi_m {\Psi_k}^2 \rangle. \tag{5.9.11}$$

Now, we can write Ψ_k in term of ψ_k and taking the expectation of the product of y and their square.

In the case of the matrix $[G_0]_{ii}$, we have $\xi_0 = 1$ and then then $[G_0]_{ii} = \langle {\Psi_i}^2 \rangle = 1$. This is by the orthogonality and hence the matrix G_0 is the identity matrix.

Double orthogonal polynomials

For the construction of the stochastic subspace, we choose the order of (KL) expansion ($M = 4$) and we chose the total degree of the orthogonal Legendre polynomial $p = 2$. The first $(p + 1)$ three orthogonal Legendre polynomial of degree 2 or less are

$$P_0(y) = 1, \quad P_1(y) = y, \quad P_2(y) = \frac{1}{2}(3y^2 - 1). \tag{5.9.12}$$

They form an orthogonal basis in $L^2(I)$ where $I = (-1, 1)$ with the weight function 1 and they satisfy

$$\int_{-1}^{1} P_n(x)P_m(x)dx = \frac{2}{(2n+1)}\delta_{nm}, \tag{5.9.13}$$

where δ_{nm} is the Kronecker delta. We also use the eigenpairs (λ_m, k_m) calculated in the subsection 5.10.1.

Here we construct the double orthogonal polynomials to span the tensor product space $\Psi_p(\Gamma)$ that approximates $L^2_\rho(\Gamma)$.

To this end, let C_j $(1 \leq j \leq p+1 = 3)$, be the three roots of the ρ–orthogonal Legendre polynomial $\frac{1}{2}(5y^3 - 3y = 0)$ where ρ is the p.d.f. of the Uniformly distribution.

At the beginning, we find the Lagrange polynomials basis $L_1(y), L_2(y)$ and $L_3(y)$ associated with these three roots such that they satisfy $L_j(y_k) = \delta_{jk},\ k = 1, 2, 3$. Let $\{\psi_i\}_{i=1}^{3}$ denotes the set of the double orthogonal polynomials (which are resulted from the Lagrange polynomials on the three nodes (roots))

$$C_1 = 0, \quad C_2 = \sqrt{\frac{3}{5}}, \quad C_3 = -\sqrt{\frac{3}{5}} \tag{5.9.14}$$

Then as we know

$$\begin{aligned} L_1(y) &= \alpha_1 \frac{(y - C_2)(y - C_3)}{(C_1 - C_2)(C_1 - C_3)}, \\ L_2(y) &= \alpha_2 \frac{(y - C_1)(y - C_3)}{(C_2 - C_1)(C_2 - C_3)}, \\ L_3(y) &= \alpha_3 \frac{(y - C_1)(y - C_2)}{(C_3 - C_2)(C_3 - C_2)}. \end{aligned} \tag{5.9.15}$$

To find $\alpha_i,\ i = 1, 2, 3$, one can use the orthogonality condition. Now, let

$$\begin{aligned} \psi_1(y) &= \frac{3}{2} L_1(y) = \frac{-5}{2}(y^2 - \frac{3}{5}), \\ \psi_2(y) &= \sqrt{\frac{18}{5}} L_2(y) = \sqrt{\frac{5}{2}} y (y + \sqrt{\frac{3}{5}}), \\ \psi_3(y) &= \sqrt{\frac{18}{5}} L_3(y) = \sqrt{\frac{5}{2}} y (y - \sqrt{\frac{3}{5}}). \end{aligned} \tag{5.9.16}$$

Note that when Hermite polynomial of degree 2 or less are used (which is not our case), the corresponding double orthogonal polynomials can be written as

$$\begin{aligned} \psi_1(y) &= \frac{L_1(y)}{\sqrt{\frac{2}{3}}} = \frac{3-y^2}{\sqrt{6}}, \\ \psi_2(y) &= \frac{L_2(y)}{\sqrt{6}} = \frac{y(y+\sqrt{3})}{\sqrt{6}}, \\ \psi_3(y) &= \frac{L_3(y)}{\sqrt{6}} = \frac{y(y-\sqrt{3})}{\sqrt{6}}. \end{aligned} \tag{5.9.17}$$

Note that the above set $\{\psi_i\}_{i=1}^3$ given in (5.9.16) satisfies the conditions in (5.8.13) and any element in the set $\{\Psi_i\}_{i=1}^{N_\xi=81}$ can be written as a product of those polynomials $\{\psi_i\}_{i=1}^3$ i.e. for $k = 1, ..., 81$, $\Psi_k = \prod_{j=1}^{M=4} \psi_j(y_j)$ (see [33] for more details). Note that the first three orthogonal Legendre polynomial on $[-\sqrt{3}, \sqrt{3}]$ with weighted function $\frac{1}{2\sqrt{3}}$ are

$$P_0(y) = 1, \quad P_1(y) = y, \quad P_2(y) = \frac{\sqrt{5}}{2}(y^2 - 1). \tag{5.9.18}$$

Then, the corresponding double orthonormal polynomials can be written as

$$\begin{aligned} \psi_1(y) &= \frac{-5}{6}(y^2 - \frac{9}{5}), \\ \psi_2(y) &= \sqrt{\frac{5}{18}} y(y + \frac{3}{\sqrt{5}}), \\ \psi_3(y) &= \sqrt{\frac{5}{18}} y(y - \frac{3}{\sqrt{5}}). \end{aligned} \tag{5.9.19}$$

The advantage of the construction of the double orthogonal polynomials is to decouple the resulting huge linear system (5.8.2) of size $N_\xi(N_u + N_p) = 81(1024 + 545)$ into 81 saddle point problems each of which of size $(1024 + 545)$.

Remark 8

The Matlab-codes for the computation of this chapter can be found in Chapter 8.

5.10 Conclusions

In this chapter, we introduce a new stochastic formulation for mixed Darcy's equations. This formulation leads to reduction of the computations's cost. Analysis of the discretization of this formulation is presented. Moreover, we use the double-orthogonal basis to the stochastic space. This basis leads to digitalized the stochastic matrices. This property leads to decouple the system. In other words, instead of solving a single large system, we solve decoupled systems of small sizes.

Chapter 6

PRECONDITIONING TECHNIQUE FOR STOCHASTIC DARCY'S EQUATIONS

6.1 Introduction

In this chapter, we consider the following decoupled linear system

$$\underbrace{\begin{bmatrix} A^i & B^T \\ B & 0 \end{bmatrix}}_{C^i} \begin{bmatrix} \mathbf{u}^i \\ \mathbf{p}^i \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}^i \end{bmatrix}. \qquad (6.1.1)$$

Where A^i given by

$$A^i = A_0 + G(1,i)A_1 + ... + G(M,i)A_M, \quad i = 1, ..., N_\xi, \qquad (6.1.2)$$

and $G(M,i) = [G_M]_{ii}$. As we mentioned in the previous chapter, the above system results from discretization Darcy's equations with stochastic coefficients after permuting the N_ξ blocks of unknowns as in (5.8.3). This decoupling property is resulted of using the so called double orthogonal polynomials as a basis for the stochastic subspace Ψ_p.

So, we have N_ξ saddle point system each of which of size $N_u + N_p$. The first saddle point system is in the form

$$\underbrace{\begin{bmatrix} A^1 & B^T \\ B & 0 \end{bmatrix}}_{C^1} \begin{bmatrix} \mathbf{u}^1 \\ \mathbf{p}^1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}^1 \end{bmatrix}. \qquad (6.1.3)$$

where $A^1 = A_0 + G(1,1)A_1 + ... + G(M,1)A_M$ and so on. In the previous chapter, we solved the decoupled system by using **the built in MATLAB- linear solver** (slash solution). In this chapter, we propose several preconditioners to solve the N_ξ-linear systems. These preconditioners are based on block-diagonal preconditioners for the deterministic saddle-point system (5.5.5). Moreover we study the eigenvalues bound of their preconditioned matrices. Finally, we mach the driven bounds of the eigenvalues with some numerical examples. Hence, it is better to start by preconditioning the deterministic Problem.

6.2 Deterministic problem

Consider the following linear saddle point system

$$\underbrace{\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}}_{\text{C}} \begin{bmatrix} \underline{u} \\ \underline{p} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{f} \end{bmatrix}, \tag{6.2.1}$$

The above saddle point linear system results from the $[L^2(D)]^2 \times (H^1(D) \cap L_0^2(D))$ finite element method (FEM) for Darcy's equations.

The advantage of using this formulation is that the (1,1)-block of the coefficient matrix of the system (6.2.1) is diagonal and the Schur complement matrix, $S = BA^{-1}B^T$, is the discrete analog of the Laplace operator $\mathbf{L} = -(k\nabla \cdot \nabla)$ where k is the permeability coefficient. These advantages lead to an efficient solution to the problem. In this section, we introduce an efficient preconditioner for the system (6.2.1). The advantage of the preconditioner is to give a fast rate of convergence of the used iterative method. This preconditioner has two nice properties. The first property is that the (1,1) block is a diagonal matrix while the second one is that the (2,2) block is the well known discrete Laplacian matrix. The diagonal property leads to cheap inversion and the well known Schur complement suggests using algebraic multigrid (AMG) [74]. Also, another block diagonal preconditioner is presented. This preconditioner comes from the matrices representation of the discrete velocities and pressure norms. This preconditioner is called the natural preconditioner [79]. In both preconditioners, we use the minimal residual method (MINRES) as an iterative solver.

6.2.1 Preconditioners for the deterministic problem

To construct our preconditioners, we need to introduce the discrete representations of the norms in term of matrices. Before doing this task, let us define the velocity mass matrix $M \in \mathbb{R}^{n\times n}$, the velocity weight mass matrix $A \in \mathbb{R}^{n\times n}$, the pressure gradient matrix $G \in \mathbb{R}^{m\times m}$, the pressure mass matrix $N \in \mathbb{R}^{m\times m}$ and the Laplace scaling matrix $L \in \mathbb{R}^{m\times m}$ as follows:

$$\begin{aligned} M_{i,j} &= (\varphi_i, \varphi_j), & 1 \le i,j \le n. \\ A_{i,j} &= (k^{-1}\varphi_i, \varphi_j), & 1 \le i,j \le n. \\ G_{i,j} &= (\nabla\phi_i, \nabla\phi_j), & 1 \le i,j \le m. \\ N_{i,j} &= (\phi_i, \phi_j), & 1 \le i,j \le m. \\ L_{i,j} &= -(k\nabla\phi_i, \nabla\phi_j), & 1 \le i,j \le m. \end{aligned} \tag{6.2.2}$$

Now for any $v_h \in X_h$ and $q_h \in Q_h$, the discrete representations of the norms can be written as

$$\begin{aligned} \| \vec{v}_h \|_{L^2} &= \underline{v}^T M \underline{v}, \\ \| \vec{q}_h \|_{L^2} &= \underline{q}^T N \underline{q}, \\ \| \vec{q}_h \|_{H^1} &= \underline{q}^T (N+G) \underline{q}, \end{aligned} \tag{6.2.3}$$

The proposed preconditioners are given as

$$P_1 := \begin{bmatrix} A & 0 \\ 0 & L_{AMG} \end{bmatrix} \quad P_2 := \begin{bmatrix} A & 0 \\ 0 & N+G \end{bmatrix}, \tag{6.2.4}$$

In (6.2.4), L_{AMG} represents the action of algebraic multigrid cycles applied to the Poisson problem. The main key in the first preconditioner is the relationship between

B, L and A. This relationship is given in the following lemma.

Lemma 1 $L = BA^{-1}B^T$. ***Proof:*** *Consider the following operators*

$$\boldsymbol{B} : X_h \to Q_h,\ \boldsymbol{M} : X_h \to X_h,\ \boldsymbol{A} : X_h \to X_h,\ \boldsymbol{L} : Q_h \to Q_h, \tag{6.2.5}$$

and consider the following matrices in the operator forms. That is,

$$\begin{aligned}
(\mathbf{B}x_h, z_h) &= (\nabla \cdot x_h, z_h),\ \forall x_h \in X_h,\ \forall z_h \in Q_h,\\
(\mathbf{B}^T z_h, x_h) &= (\nabla z_h, x_h),\ \forall x_h \in X_h,\ \forall z_h \in Q_h,\\
(\mathbf{A}x_h, y_h) &= (k^{-1}x_h, y_h) = (x_h, k^{-1}y_h),\ \forall x_h,\ y_h \in X_h,\\
(\mathbf{L}z_h, z_h) &= -(k\nabla z_h, \nabla z_h),\ \forall z_h \in Q_h,\\
(\mathbf{M}x_h, y_h) &= (x_h, y_h),\ \forall x_h,\ y_h \in X_h,\\
(\mathbf{M}k^{-1}x_h, y_h) &= (k^{-1}x_h, y_h) = (\mathbf{A}x_h, y_h),\ \forall x_h,\ y_h \in X_h,
\end{aligned} \tag{6.2.6}$$

Now, with using the above defections and using $(\nabla \cdot x_h, z_h) = -(x_h, \nabla z_h),\ \forall x_h \in X_h,\ \forall z_h \in Q_h$, *we have*

$$\begin{aligned}
(\mathbf{L}z_h, z_h) &= -(k\nabla z_h, \nabla z_h)\\
&= -(\nabla z_h, k\nabla z_h)\\
&= -(\mathbf{B}^T z_h, k\nabla z_h)\\
&= -(\mathbf{B}^T z_h, \boldsymbol{M}k\nabla z_h)\\
&= (\mathbf{B}\boldsymbol{A}^{-1}\mathbf{B}^T z_h, z_h),
\end{aligned} \tag{6.2.7}$$

which proves the above lemma.

6.2.2 Eigenvalue analysis

Murphy, Golub and Wathen proved that the eigenvalues of the preconditioned matrix $P_1^{-1}C$, independent of the mesh size h, are only three distinct eigenvalues

$$\frac{1}{2}(1-\sqrt{5}), 1, \frac{1}{2}(1+\sqrt{5}). \tag{6.2.8}$$

For the eigenvalues of $P_2^{-1}C$, we use the following theorem

Theorem 8 (Lemma 2.1 in [84]) *Let*

$$\mathbb{A} := \begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \tag{6.2.9}$$

be a symmetric, nonsingular, and indefinite matrix and let $0 < \mu_1 \leq \mu_2 \leq ... \leq \mu_n$ *be the eigenvalues of* A*,* $0 < \sigma_1 \leq \sigma_2 \leq ... \leq \sigma_m$ *the singular values of* B*, and denote by* $\Lambda(\mathbb{A})$ *the spectrum of* $\mathbb{A}$*. Then* $\Lambda(\mathbb{A}) \subset I \equiv I^- \cup I^+$ *where*

$$\begin{aligned} I^- &= [\frac{1}{2}(\mu_1 - \sqrt{\mu_1^2 + 4\sigma_m^2}), \frac{1}{2}(\mu_n - \sqrt{\mu_n^2 + 4\sigma_1^2})], \\ I^+ &= [\mu_1, \frac{1}{2}(\mu_n + \sqrt{\mu_n^2 + 4\sigma_m^2})]. \end{aligned} \tag{6.2.10}$$

Theorem 9 *The eigenvalues of* $P_2^{-1}C$ *lie in the union two intervals*

$$[\frac{1}{2}(1-\sqrt{1+4\sigma_m^2}), \frac{1}{2}(1-\sqrt{1+4\sigma_1^2})] \cup [1, \frac{1}{2}(1+\sqrt{1+4\sigma_m^2})], \tag{6.2.11}$$

where σ_1 and σ_m are the smallest and largest singular values of the matrix $(N + G)^{\frac{-1}{2}} B A^{\frac{-1}{2}}$.

Proof: We start expressing the conditioned matrix $P_2^{-1}C$ in a generalized saddle point matrix. $P_2^{-1}C$ is similar to $P_2^{\frac{1}{2}}(P_2^{-1}C)P_2^{\frac{-1}{2}} = P_2^{\frac{-1}{2}}CP_2^{\frac{-1}{2}} =$

$$= \begin{bmatrix} A^{\frac{-1}{2}} & 0 \\ 0 & (N+G)^{\frac{-1}{2}} \end{bmatrix} \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} A^{\frac{-1}{2}} & 0 \\ 0 & (N+G)^{\frac{-1}{2}} \end{bmatrix} \tag{6.2.12}$$

$$= \begin{bmatrix} A^{\frac{1}{2}} & A^{\frac{-1}{2}}B^T \\ (N+G)^{\frac{-1}{2}} & 0 \end{bmatrix} \begin{bmatrix} A^{\frac{-1}{2}} & 0 \\ 0 & (N+G)^{\frac{-1}{2}} \end{bmatrix} \tag{6.2.13}$$

$$= \begin{bmatrix} I & A^{\frac{-1}{2}}B^T(N+G)^{\frac{-1}{2}} \\ (N+G)^{\frac{-1}{2}}BA^{\frac{-1}{2}} & 0 \end{bmatrix} \tag{6.2.14}$$

$$= \begin{bmatrix} I & \tilde{B}^T \\ \tilde{B} & 0 \end{bmatrix} = \tilde{\mathcal{A}} \tag{6.2.15}$$

Now using Theorem 8, one can obtain the results.

6.2.3 Numerical computations

In this section, we investigate the efficiency of the two preconditioners P_1 and P_2 given in (6.2.4) via several computations. We solve the saddle point system (6.2.1) by PMINRES method and observe the iteration numbers. It is known that in each PMINRES iteration, we solve a linear system of the form $Px = y$. To solve this system, we use the black-box AGgregation-based algebraic MultiGrid (AGMG) solver for the (2,2) block (see [74] for details).

Example 7 *Here, we consider the well known five spot problem (5.1.5) in the domain* $D = [0,1] \times [0,1]$. *In this problem, we place an injection well at the center of the domain and production wells at the corners and specify no-flow conditions at the*

h, iter. NO.	h=1/4	h=1/8	h=1/16	h=1/64
P_2	10	8	6	4
P_1	2	2	2	2

Table 6 .1: $k^{-1} \equiv 1$

h, iter. NO.	h=1/4	h=1/8	h=1/16	h=1/64
P_2	20	18	16	14
P_1	2	2	2	2

Table 6 .2: $k^{-1} = 1 + x^2 + y^2$

boundaries. In other word we define the data function f as follows

$$f(x,y) = \begin{cases} 1, & \textit{if (x,y)=}\{\textit{(0,0),(1,0),(0,1),(1,1)}\}; \\ -1, & \textit{if (x,y)=}\{\textit{(1/2,1/2)}\}; \\ 0, & \textit{otherwise.} \end{cases} \tag{6.2.16}$$

We solve the resulting linear system (6.2.1) using PMINRES with P_1 and P_2 as preconditioners. We chose the tolerance to be $1e-8$ *and we record the iteration numbers for different meshsizes and different coefficient k^{-1}. The iteration numbers are tabulated in Tables (6.1-6.3). Table 6.1 is obtained when $k^{-1} \equiv 1$, Table 6.2 when $k^{-1} \equiv 1 + x^2 + y^2$ and Table 6.3 when $k^{-1}(x) = \exp(x) + \exp(y)$. We plot the logarithm of the L^2 norm of the ratio $\frac{\|r^{(k)}\|_2}{\|r^{(0)}\|_2}$ .vs. the iteration numbers for different mesh size and when $k^{-1}(x) = \exp(x) + \exp(y)$ see Figures (6.4-6.7) From Tables (6.1-6.3), we observe that the number of PMINRES iterations by using P_1 are less*

h, iter. NO.	h=1/4	h=1/8	h=1/16	h=1/64
P_2	22	20	18	16
P_1	2	2	2	2

Table 6 .3: $k^{-1}(x) = \exp(x) + \exp(y)$

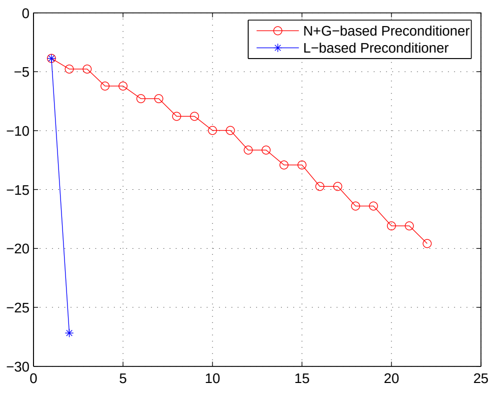


Figure 6 .1: when $h = 1/4$

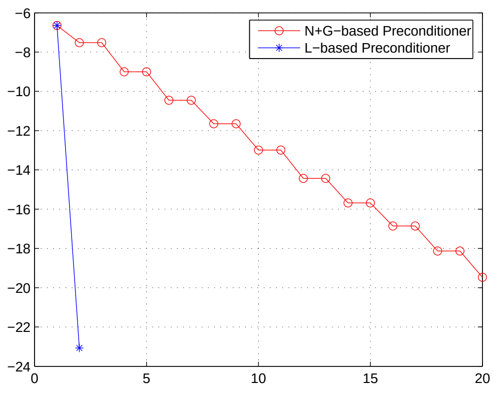


Figure 6 .2: when $h = 1/8$

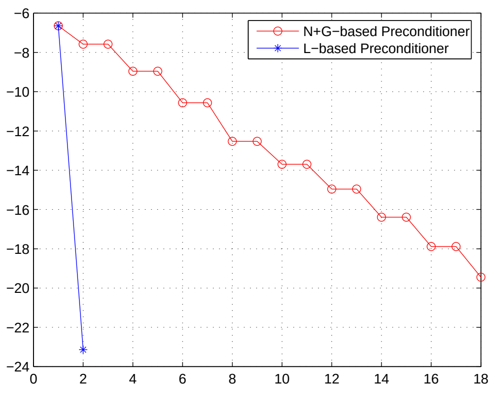


Figure 6 .3: when $h = 1/16$

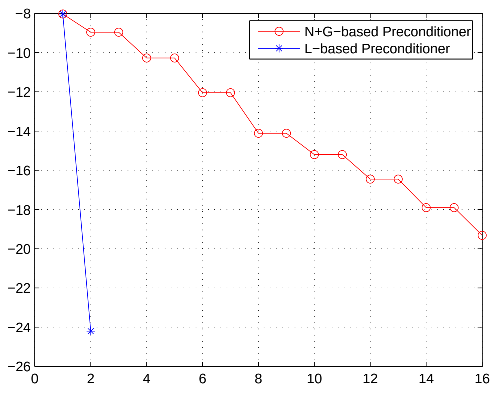


Figure 6 .4: when $h = 1/64$

h	$k^{-1}(x) \equiv 1)$	$k^{-1}(x) = 1 + x^2 + y^2$	$k^{-1}(x) = \exp(x) + \exp(y)$
$1/4$	2	2	2
$1/8$	2	2	2
$1/16$	2	2	2
$1/64$	2	2	2

Table 6 .4: P_1MINRES iterations NO.

than the number of PMINRES iterations by using P_2. Hence, PMINRES with using P_1 as a preconditioner is more efficient.

In the next two examples, we only consider the first preconditioner P_1.

Example 8 *In this example we show that the number of PMINRES iterations with P_1 are independent of the mesh size and the permeability coefficient. We solve the same problem as in Example 8 with varying the mesh sizes and the permeability $k^{-1}(x)$ and list the number of iterations in Table 6.4.*

Example 9 *In this example we also solve the five spot problem by PMINRES and P_1 as a preconditioner in the domain $D = [0,1] \times [0,1]$ with data function f defined as above. The shape of the used mesh (16384 triangles and 8321 nods), the pressure and the velocities are plotted in Figures (6.8-6.13).*

6.3 Preconditioners for the decoupled stochastic system

In the following, we shall construct preconditioners to the stochastic Galerkin equations (6.1.1) based on the deterministic preconditioners given in the above section. First, we use the following preconditioner:

6.3.1 Laplace preconditioner

$$P_L = \begin{pmatrix} A_0 & 0 \\ 0 & S_0 = L_0 \end{pmatrix}$$

where $L_0 = BA_0^{-1}B^T$.

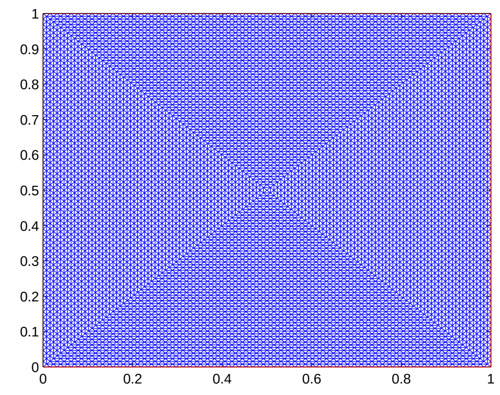


Figure 6 .5: Shape of the mesh

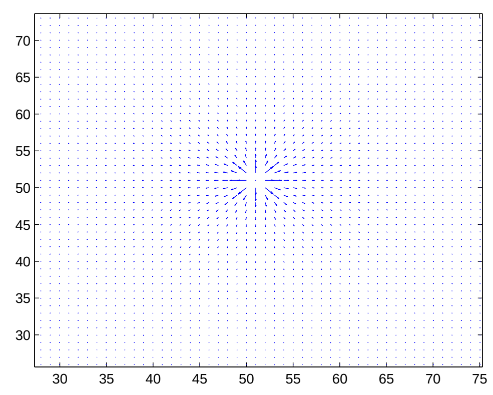


Figure 6 .6: Velocity distribution

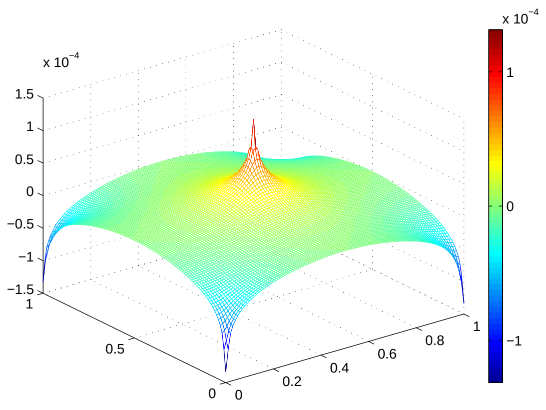


Figure 6 .7: Pressure surface

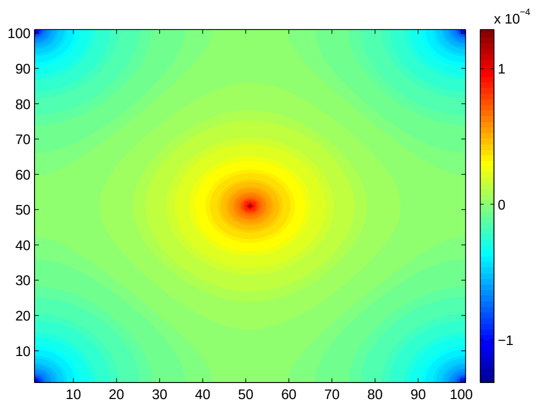


Figure 6 .8: Pressure cantor

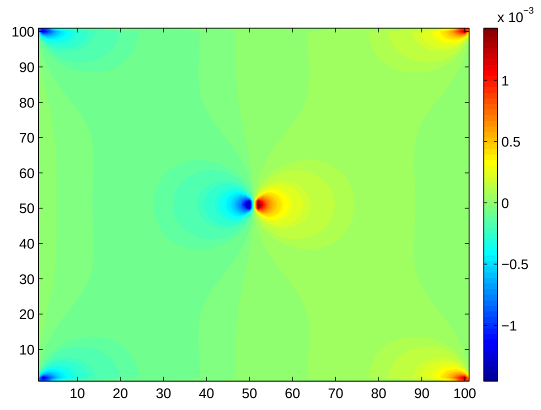


Figure 6 .9: The u^x velocity

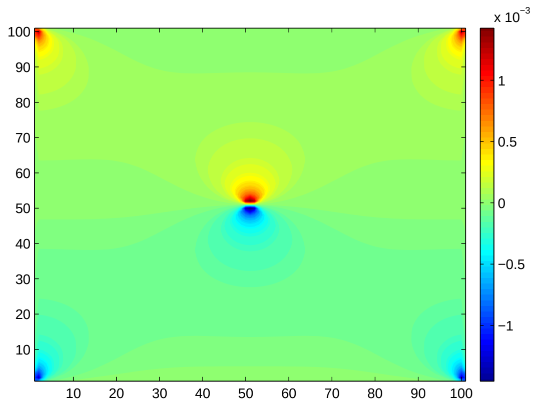


Figure 6 .10: The u^y velocity

Eigenvalues bounds of $P_L^{-1}C^i$

Theorem 10 *The eigenvalues of $P_L^{-1}C^i$ lie in the union two intervals*

$$[\frac{1}{2}(\lambda^i_{min} - \sqrt{{\lambda^i_{min}}^2 + 4}), \frac{1}{2}(\lambda^i_{max} - \sqrt{{\lambda^i_{max}}^2 + 4})] \cup [\lambda^i_{min}, \frac{1}{2}(\lambda^i_{max} + \sqrt{{\lambda^i_{max}}^2 + 4})], \tag{6.3.1}$$

where λ^i_{min} and λ^i_{max} are the smallest and largest eigenvalue of the matrix $A_0^{-1}A^i$, $i = 1, ..., N_\xi$.

Proof: We start expressing the conditioned matrix $P_L^{-1}C^i$in a generalized saddle point matrix. $P_L^{-1}A^i$ is similar to $P_L^{\frac{-1}{2}}C^iP_L^{\frac{-1}{2}} =$

$$= \begin{bmatrix} A_0^{\frac{-1}{2}} & 0 \\ 0 & L_0^{\frac{-1}{2}} \end{bmatrix} \begin{bmatrix} A_i & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} A_0^{\frac{-1}{2}} & 0 \\ 0 & L_0^{\frac{-1}{2}} \end{bmatrix} \tag{6.3.2}$$

$$= \begin{bmatrix} A_0^{\frac{-1}{2}}A_i & A_0^{\frac{-1}{2}}B^T \\ L_0^{\frac{-1}{2}}B & 0 \end{bmatrix} \begin{bmatrix} A_0^{\frac{-1}{2}} & 0 \\ 0 & L_0^{\frac{-1}{2}} \end{bmatrix} \tag{6.3.3}$$

$$= \begin{bmatrix} A_0^{\frac{-1}{2}}A_iA_0^{\frac{-1}{2}} & A_0^{\frac{-1}{2}}B^TL_0^{\frac{-1}{2}} \\ L_0^{\frac{-1}{2}}BA_0^{\frac{-1}{2}} & 0 \end{bmatrix} \tag{6.3.4}$$

$$= \begin{bmatrix} A_0^{-1}A_i & \tilde{B}^T \\ \tilde{B} & 0 \end{bmatrix} = \tilde{\mathcal{A}} \tag{6.3.5}$$

Now using Theorem 9, one can obtain the desired result. We use the MINRES with the above preconditioner as a linear solver. We generate a small mesh contains 16 triangles with 13 nodes to compute the eigenvalues of the preconditioned matrix. We set the maximum iteration =100 of the linear solver and the tolerance =1e-5. Here, we chose M=4 and P=2 and we use the Uniform distribution with Lagender polynomial. The computation results are listed in Tables (6.5) and (6.7).

i	Iter	Bounds	Computed eigenvalues
1	8	[-0.6180, -0.6180] ∪ [1, 1.6180]	[-0.6180, -0.6180] ∪ [1, 1.6180]
10	11	[-0.6360, -0.6008]∪[0.9363, 1.6645]	[-0.6357, -0.6011] ∪[0.9373, 1.6637]
20	9	[-0.6417, -0.6060]∪[0.9168, 1.6502]	[-0.6413, -0.6063]∪[0.9179, 1.6495]
30	9	[-0.5776, -0.5679]∪[1.1539, 1.7607]	[-0.5776, -0.5679]∪[1.1539, 1.7607]
40	10	[-0.5947, -0.5519]∪[1.0867, 1.8119]	[-0.5920, -0.5525]∪[1.0940, 1.8100]
50	7	[-0.5958, -0.5488]∪[1.0825, 1.8221]	[-0.5913, -0.5522]∪[1.0938, 1.8114]
60	9	[-0.6954, -0.6550]∪[0.7427, 1.5267]	[-0.6946, -0.6557]∪[0.7450, 1.5251]
70	9	[-0.6943, -0.6445]∪[0.7459, 1.5516]	[-0.6936, -0.6474]∪[0.7461, 1.5446]
80	8	[-0.7027, -0.6398]∪[0.7205, 1.5629]	[-0.6977, -0.6454]∪[0.7294, 1.5500]
81	8	[-0.6995, -0.6370]∪[0.7302, 1.5698]	[-0.6957, -0.6415]∪[0.7349, 1.5592]

Table 6 .5: Bounds on the eigenvalues of $P_L^{-1}C^i$

6.3.2 Natural preconditioner

Here, we use the Natural preconditioner

$$P_N = \begin{pmatrix} A_0 & 0 \\ 0 & N+M \end{pmatrix}$$

given in above section.

Eigenvalues bounds of $P_N^{-1}C^i$

Theorem 11 *The eigenvalues of $P_N^{-1}C^i$ lie in the union two intervals*

$$[\frac{1}{2}(\lambda^i_{min} - \sqrt{{\lambda^i_{min}}^2 + 4{\nu_n}^2}), \frac{1}{2}(\lambda^i_{max} - \sqrt{{\lambda^i_{max}}^2 + 4{\nu_1}^2})] \cup [\lambda^i_{min}, \frac{1}{2}(\lambda^i_{max} + \sqrt{{\lambda^i_{max}}^2 + 4{\nu_n}^2})], \tag{6.3.6}$$

where ν_1 and ν_n are the smallest and largest eigenvalue of the matrix $(N+M)^{-1}L_0$ and λ^i_{min} and λ^i_{max} are the smallest and largest eigenvalue of the matrix $A_0^{-1}A^i$, $i = 1, ..., N_\xi$. The proof is similar to the above proofs. We also use the same numerical example given above. The results of this computation are listed in Tables (6.6) and

i	Iter	Bounds	Computed eigenvalues
1	8	[-0.6154 -0.5290] ∪ [1.0000 1.6154]	[-0.6154, -0.5290] ∪ [1.0000 1.6154]
10	11	[-0.6333 -0.5130]∪[0.9363 1.6619]	[-0.6304, -0.5288] ∪[0.9373, 1.6573]
20	8	[-0.6390 -0.5178]∪[0.9168 1.6476]	[-0.6354, -0.5288]∪[0.9179, 1.6444]
30	8	[-0.5750 -0.4828]∪[1.1539 1.7582]	[-0.5728, -0.4841]∪[1.1539, 1.7561]
40	9	[-0.5921 -0.4681]∪[1.0867, 1.8094]	[-0.5873, -0.4838]∪[1.0940, 1.8049]
50	7	[-0.5932 -0.4653]∪[1.0825 1.8196]	[-0.5861, -0.4839]∪[1.0938, 1.8050]
60	8	[-0.6926 -0.5636]∪[0.7427 1.5240]	[-0.6887, -0.5801]∪[0.7450, 1.5190]
70	8	[-0.6916 -0.5537]∪[0.7459 1.5489]	[-0.6886, -0.5799]∪[0.7461, 1.5391]
80	8	[-0.6999 -0.5494]∪[0.7205 1.5602]	[-0.6920, -0.5798]∪[0.7293, 1.5435]
81	8	[-0.6967 -0.5467]∪[0.7302 1.5671]	[-0.6905, -0.5798]∪[0.7349, 1.5519]

Table 6 .6: Bounds on the eigenvalues of $P_N^{-1}C^i$

i	λ^i_{max}	λ^i_{min}
1	1	1
10	1.0637	0.9363
20	1.0443	0.9168
30	1.1928	1.1539
40	1.2600	1.0867
50	1.2732	1.0825
60	0.8717	0.7427
70	0.9071	0.7459
80	0.9231	0.7205
81	0.9328	0.7302

Table 6 .7: Maximum and minimum eigenvalues of $A_0^{-1}A^i$

(6.7).

6.3.3 Exact Schur complement preconditioner

Here, we use the following preconditioner

$$P_S = \begin{pmatrix} A^i & 0 \\ 0 & S^i = BA_i^{-1}B^T, \end{pmatrix}$$

where A^i is given in (6.1.2). In this case, the eigenvalues of the preconditioned matrix $P^{-1}C_i$ are only three 1, $\frac{(1-\sqrt{5})}{2}$ and $\frac{(1+\sqrt{5})}{2}$ and PMINRES needs only 2 iterations to reach the solution with the same tolerance as in the above two examples.

6.4 Conclusion

We solved the decoupled systems given in Chapter 5 by using preconditioner technique. We present three preconditioners for the deterministic problem and test their performance in several numerical examples. Moreover, we study the eigenvalues analysis of their preconditioned matrices. Finally, we purpose also three block diagonal preconditioners for the stochastic problem. These preconditioners are based on the preconditioners of the deterministic problem. We study the bounds of the eigenvalues and match these bound by numerical examples. Moreover, we examine the performance of these preconditioners through several examples.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this dissertation, we consider two saddle point system of equations. The first system arises when the total variational regularization is applied to solve an ill-posed problem (image deblurring problem) while the second system results from the discretization of the $([L^2(D)]^2 \times L^2_{\mathbb{P}}(\Omega)) \times (H^1(D) \cap L^2_0(D) \times L^2_{\mathbb{P}}(\Omega))$ formulation for the stochastic Darcy's equations. These system are huge and ill-conditioned. Hence, the numerical solutions to these system represent a big challenge. This challenge attract us to work with these system. In chapter 3, we use the total variation as a regulazation term to stable the minimization of the image deblurring problem. This type of the regulazation is not easy to compute but it gives a good result. In other words, the reconstruction image has no ringing or oscillation as in the Tikhanov regulazation. We also introduce the mathematical model behind image deblurring problems. The main our contributions in this chapter is that this is the first studies for the mixed formulation of the image deblurring problem. In Chapter 4, we propose several preconditioners which is the main our contribution in this chapter. Moreover, we study the eigenvalues bound of their preconditioned matrices and mach the theoretical results by numerical examples. The proposed preconditioners depend on the circulant matrices. This circulant matrices allow us to use the fast Fourier transform to do the matrix-vector multiplications. This transform reduces the cost of the computations and also the storage. Moreover, we compare between the preconditioners through several numerical examples. The last contribution in this chapter is that we use two positive parameters to enforce the clustering behavior of the eigenvalues and then to have a convergence with few iterations.

For the second system of equations which is resulted from stochastic Darcy's equations, in Chapter 5, we use the well known stochastic Galrkin finite element method. In this method we use the standared finite element method to discretized the spatial space while the tensor polynomial spaces is used to generate a basis for the stochastic space. We also expand the random field by using the well known KL-expansion. In this expansion, the random field is written as a summation of product of two functions one is deterministic and the second is a random variable with scalar terms come from the spectral analysis of the KL-expansion. The mean our contribution in this Chapter is that we introduce a new formulation and study the analysis of this formulation (existence, uniqueness and error analysis). This formulation leads to more reduction of the compactions because the (1,1)-block matrix in the coefficient matrix of the saddle point system is diagonal. This diagonally property leads to easy inverse computations. Finally, we solve the deterministic and stochastic examples (Five-spot problem). In Chapter 6, we start showing that the Shur complement of the deterministic problem is the Laplace operator. Moreover, we introduce more effective preconditioners for both deterministic and stochastic problem which are the main our contributions in this chapter. These prconditioners have the diagonally structure property of their (1,1)-block matrix. We also studied the eigenvalues analysis of all preconditioners. Finally, we test their performance through several example.

7.2 Future works

The present study opens many possible future directions

- The image deblurring problem can be studied with different regulazation term.
- The image deblurring problem can be studied without given kernel (blind deblurring)

- The image deblurring problem can be studied with using color images.
- The image deblurring problem can be studied with using finite element methods instead of finite difference methods
- The image deblurring problem can be studied with using different boundary conditions.
- The image deblurring problem can be studied with using two-level finite element methods.
- The stochastic problem can be studied in which the right hand side is random field also.
- The stochastic problem can be studied with using wavelet basis or complete basis instead of tensor product basis.
- The stochastic problem can be studied in which the random field T is in the second and right hand side terms not in the first term.
- The stochastic problem can be studied with using modified Gaussian random field instead of uniform.
- The stochastic problem can be studied by using two-level decoupling the coupled system instead of using double orthogonal basis.
- The stochastic problem can be studied by using two-level method for reducing the order of the used polynomials.
- The stochastic problem can be studied by using two-level method for deterministic the stochastic problem.

Chapter 8

MATLAB CODES

```
%%This code uses to remove the blurry from  digital  images.
%%Through this code, several functions are used.
%%These functions are written below of the code.
clear, close all
u_exact = double(imread('retinal.PNG'));
%% double(imread) is used to read images as a matrix.
u_exact=u_exact(1:512,1:512,1); %% resize if it is not square.
N=size(u_exact,1); kernel=ke_gen(N,300,10);
%% This function generates a certain kernel  or PSF
%%  with   radius=10 and tau=300.
n =256; %% Resize the orginal image, the kernel to reduce the problem.
u_exact=imresize(u_exact,[n n]);
kernel=imresize(kernel,[n n]);
nx= n; ny = n; hx = 1 / nx; hy = 1 / ny; N=n;
%%  Extend kernel to be of size the original image and compute its 2-d FT.
%% Then use this to  compute K'*z and K*k.
kernel=kernel/sum(kernel(:)); m2 = 2*n; nd2 = n /2;
kernele=zeros(m2, m2); kernele(nd2+1:n+nd2,nd2+1:n+nd2) = kernel;
%% extension the kernel. fftshift(X) swaps the first quadrant with the third
%% and the second quadrant with the fourth.
k_ext = fftshift(kernele); k_hat = fft2(fftshift(kernele));

clear kernele %% To reduse the storages
beta =0.01;  alpha = 0.00008;  n = nx^2;  m = 2*nx*(nx-1); nm =
n+m; gamma1 = 1;  gamma2 = 1; gamma0=1; %% The input parameters

computeB;    U = zeros(nx,nx);  [D] = computeD(U,nx,m,beta);
```

```
L=B'*inv(D)*B; figure;
imagesc(u_exact);s=sprintf('exactimage');s=title(s);colormap(gray);
%% imagesc is used to plot data from a 2-D matrix.
z = integral_ke(u_exact,k_hat,nx,nx);  % Blur the exact image
figure;
imagesc(z);ss=sprintf('bluredimage');ss=title(ss);colormap(gray)
zv = z(:);   b2 = integral_ke(z,conj(k_hat),nx,nx);
b=[sparse(m,1); -b2(:) ]; %% The right hand side
% -----------------------------------
xprecond =0; %% zero if MINRES  with out preconditioner is used
%% or =1 with preconditioner.
maxit = 500;  tol=1e-2; tolconjgrad = 1e-2;

u0 = zeros(length(b),1); %%or you can take z as initial data.
t=restrict(ifft2(abs(fft2(k_ext)).^2));
t_ext=(embed((embed(t))'))'; t_ext_hat=fft2(t_ext);
u1=zeros(nx,nx); u1(1,1)=1; row1_K=integral_ke(u1,k_hat',nx,nx);
col1_K=integral_ke(u1,k_hat,nx,nx);
%% -----------------------------------
%%- If You Need To Use The Preconditioner P_T Do The Following-
rowcol=[row1_K;col1_K];
%% --------Then Use MINRES-------
%%------- If You Need To Use The Preconditioner P_S Do The Following-
row1=reshape(row1_K,nx,nx); col1=reshape(col1_K,nx,nx); c=[];

for k=1:nx; t=[col1(nx:-1:2,k)',row1(:,k)']; [s]=strang_cir(t);
c=[c,s];
```

```
end

c=reshape(c,nx,nx); c=fft2(c); rowcol=c;

%% ------Then Use MINRES------

%%------- If You Need To Use The Preconditioner P_c Do The Following-

row1=reshape(row1_K,nx,nx); col1=reshape(col1_K,nx,nx); c=[];


for k=1:nx; t=[col1(nx:-1:2,k)',row1(:,k)']; s=optimal_circ(t);

c=[c,s];

end

%%--------Then Use MINRES-------

[u,res,iter,flag] = pminres(nx,k_hat,alpha,B,D,b,u0,maxit,tol,...,

xprecond,gamma0,gamma1,gamma2,tolconjgrad,t_ext_hat,L,rowcol);

iter uv_np = u(m+1:m+n);    u_np = reshape(uv_np,nx,nx);

%% ---- Do Fixed Point iterations ---------

xprecond =1; no_fixed_point_iterations = 5;


for i=1:no_fixed_point_iterations

fprintf('------Fixed Point iteration %3.0f --------- \n',i)

U = u_np; [D] = computeD(U,nx,m,beta); u0 = u;  tol=1e-2; tic

[u,res,iter,flag] = pminres(nx,k_hat,alpha,B,D,b,u0,maxit,tol...

,xprecond,gamma0,gamma1,gamma2,tolconjgrad,t_ext_hat,L,rowcol);

toc iter


if i==2 resT=res; save resT resT %%  to plot the resduals

end


uv_np = u(m+1:m+n); u_np = reshape(uv_np,nx,nx);
```

```
psnrv(i)=psnr(u_np,u_exact); figure

imagesc(u_np);ss=sprintf('deblured image');ss=title(ss);

colormap(gray)

end

%% Plotting the deblurred images

figure imagesc(u_np);ss=sprintf('deblured image');

ss=title(ss);colormap(gray)

psnrv psnr(z,u_exact)

%% ----------------------ke_kernel-----------------------

%% This function is used to generate a certain kernel

function K = ke_kernel(n, tau, radi);

if nargin<1,help

ke_gen;return; end if nargin<2, tau=200; end


if nargin<3, radi=4; end

K=zeros(n); R=n/2; h=1/n; h2=h^2;

RR=radi^2;


if radi>0 for j=1:n for k=1:n v=(j-R)^2+(k-R)^2; if v <= RR,

K(j,k)=exp(-v/4/tau^2); end;end; end; sw=sum(K(:)); K=K/sw;


else radi<0 range=R-2:R+2; K(range,range)=1/25;

end

%%----------------------------------------------------

 function Ku = integral_ke(u,k_hat,nux,nuy)

[nkx,nky] = size(k_hat); n=size(u,1); Ku = real(ifft2(

((fft2(u,nkx,nky)) .* k_hat))); if nargin == 4 Ku
```

```
=Ku(1:nux,1:nuy); end
%--------------------------------------------------------
function [ t ] = restrict(s); [n,m]=size(s); nx = n/2; t =
(1:nx,1:nx);
%---------------------------------------------------------
function [ t_embed ] = embed(t)
%see 35 in the good_thesis
[nx,mx]=size(t); size(t) t_embed=zeros(nx,2*mx);
t_embed(1:nx,1:mx)=t; t_embed(1:nx,mx+1)=t(:,1); colm=[mx:-1:2];
for i=2:mx
t_embed(:,mx+i)=t(:,colm(i-1)); end
%----------------------compute B--------------------------
e = ones(nx,1); E = spdiags([0*e -1*e e], -1:1, nx, nx); E1
=E(1:nx-1,:); M1=eye(nx,nx); B1=kron(E1,M1); E2 = eye(nx); M2 =
spdiags([0*e -1*e e], -1:1, nx-1, nx); B2 = kron(E2,M2); B =
[B1;B2];
%-------------------compute D ----------------------------
function [D] = computeD(U,nx,m,beta); h0=1/nx; [X,Y] =
meshgrid(h0/2:h0:1-h0/2); nn = size(U,1); UU = sparse(nn+2,nn+2);
% we are using reflection bounday conditions
% another word, we are using normal boundary condition to be zero
UU(2:nn+1,2:nn+1) = U; UU(1,:) = UU(2,:); UU(nn+2,:) = UU(nn+1,:);
UU(:,1) = UU(:,2); UU(:,nn+2) = UU(:,nn+1);
Uxr = diff(U,1,2)/h0; % x-deriv at red points
xb = h0/2:h0:1-h0/2;   yr=xb; yb = h0:h0:1-h0;       xr=yb;
[Xb,Yb]=meshgrid(xb,yb); [Xr,Yr]=meshgrid(xr,yr); Uxb =
interp2(Xr,Yr,Uxr,Xb,Yb,'spline');
```

```
Uyb = diff(U,1,1)/h0; % y-deriv at blue points
Uyr = interp2(Xb,Yb,Uyb,Xr,Yr,'spline'); Dr = sqrt( Uxr.^2 +
Uyr.^2 + beta^2 ); Db = sqrt( Uxb.^2 + Uyb.^2 + beta^2 ); Dvr =
Dr(:);  Dvb = Db(:); Dv=[Dvr;Dvb]; ddd = [ sparse(m,1) , Dv ,
sparse(m,1) ];
D = spdiags(ddd,[-1 0 1],m,m);
%-------------compute small K ---------------------
K = sparse(n,n); for i=1:n ei = sparse(n,1); ei(i)=1; eim =
reshape(ei,nx,nx); Ke = integral_op(eim,kernel,nx,nx); K(:,i)
=Ke(:); end
%------------------------------------------------------------
function p = psnr(x,y)
d = mean( mean( (x(:)-y(:)).^2 ) ); m1 =
max( abs(x(:)) ); m2 = max( abs(y(:)) ); m = max(m1,m2); p =
10*log10( m^2/d );
%--------------------Strang Circulant----------------------
function [s]=strang_cir(t); length(t) n=(length(t)+1)/2 m=n/2;
m1=m-1; for k=1:m-1; s(k+1)=t(k+n); end for k=m+1:n-1;
s(k+1)=t(k); end s(m+1)=0; s(1)=t(n);
%------------------Optimal Circulant-----------------------
function [c] = optimal_circ(t);
%  Compute optimal circulant approximation C to n X n matrix A.
%    C = argmin {||B - A||_fro : B is n X n circulant}
%      = circulant(c)
A = my_toeplitz(t); [m,n] = size(A); if m ~= n fprintf('\n
***Input A must be a square matrix.\n'); return end c =zeros(n,1);
c(1) = sum(diag(A)); for j=1:n-1 c(j+1) = sum(diag(A,-j)) +
sum(diag(A,n-j)); end
c = c / n;
%---------------------Topletiz matrix------------------------
function T = my_toeplitz(t)
%  Construct n X n Toeplitz matrix T from vector t of length 2n-1.
m = max(length(t)); if mod(m,2) == 0 fprintf('\n *** Length of t
must be odd.\n'); return end n = ceil(m/2); row = t(n:-1:1); col =
t(n:m);
T = toeplitz(col,row);
%------------------------Circulant matrix--------------------------
function C = circulant(c)
%  Construct n X n circulant matrix C from vector c of length n.
if min(size(c)) > 1 fprintf('\n *** Input c must be a vector.\n');
return
end
c = c(:);  %  Make c a column vector, if it isn't already.
n = length(c); row = [c(1); c(n:-1:2)];
C = toeplitz(c,row);
%----------------------------eigenvalue computations----------
clear nx=4; s=10; beta =0.01; alpha =0.00008;
%------------------------
n = nx^2; m = 2*nx*(nx-1); u_exact =
double(imread('goldhill512.png')); N=size(u_exact,1);
kernel=ke_gen(N,300,100); [kernel]= gauss_kernel(s,nx^2);
%kernel = fspecial('gaussian',7,10);
%kernel = fspecial('gaussian',600,10);
surf(kernel)
```

```
% Resize to reduce Problem
u_exact2=imresize(u_exact,[nx nx]); kernel=imresize(kernel,[nx
nx]); computeB; U = zeros(nx,nx);
%matrix_D;
computeD; computeK; A =[alpha*D , -alpha*B;-alpha*B',-K'*K];
%--------------------------------------------------------
gamma1 =1;  gamma2 =1 ; gamma3=1; %we need gamma1 to be v small
%[C] = oomputeC_BCCB(K); %if you need just BCCB
% c = K(1,:);
% C = bccb(c);
eye_n=eye(nx^2); eye_m=eye(m); L=B'*inv(D)*B;
%S = (K'*K+alpha*L);
col1=K(:,1); row1=K(1,:); t=[col1(nx^2:-1:2)',row1(:)'];
[s]=strang_cir(t); % if you need strang circulant
C = circulant(s);
%C = bccb(s);
%[c,C] = optimal_circ(K); %if you need optimal circulant
SS=(C'*C+alpha*L); SK=K'*K+alpha*L; P =[(alpha*gamma1*D),
zeros(m,n) ; zeros(n,m) ,gamma2*SS ]; Ah = P^(-1/2)*A*P^(-1/2);
vv=eig(full(Ah)); vvv=eig(full(A)); vvsort = sort(real(vv));
vvvsort = sort(real(vvv)); vsort = [vvsort,vvvsort]
vvvv=eig(full(SK)); vvvvv=eig(full(inv(SS)*SK));   zz =
zeros(length(vv),1); ro=eig(SK^(-1/2)*L*SK^(-1/2));
ro=sort(real(ro)); sigmam=max(ro); tao=max(abs(ro));
lower_positive = 1/gamma1; upper_positive=(1 +
sqrt(1+4*alpha*sigmam))/2; lower_negative=-1; upper_negative =
-1/(1+alpha*tao); [lower_positive,upper_positive]
```

```
[lower_negative,upper_negative];
boundvec=[lower_positive,upper_positive,lower_negative,upper_negative];
zzz=[0;0;0;0]; figure plot(real(vvv),zeros(length(vvv),1),'ok')
grid on hold on figure plot(real(vvsort(2:16)),zeros(15,1),'ok')
grid on figure plot(real(vv),zz,'ro') grid on hold  on
plot(boundvec,zzz,'b*') grid on hold  on figure
plot(real(vvvv),zeros(length(vvvv),1),'ok') grid on figure
plot(real(vvvvv),zeros(length(vvvvv),1),'ok') grid on
%-------------------------------MINRES------------------
function [u_j,res,iter,flag] =
pminres(nx,k_hat,alpha,B,D,b,u_jm1,MaxIter,tol,xprecond,gamma0,
gamma1,gamma2,tolconjgrad,t_ext_hat,L,rowcol);
% Algorithm 6.1: The Preconditioned Minres Method
% page 289 from wathen book
n = length(u_jm1);  iter = MaxIter;  flag = 1; v_jm1 =
sparse(n,1);  w_jm1 = sparse(n,1);  w_j = sparse(n,1);
Au_jm1=Ax(u_jm1,nx,k_hat,alpha,B,D); %fix matrix vector
v_j = b - Au_jm1';
% use preconditioner
[z_j] = precond(xprecond,v_j,D,B,gamma0,gamma1,gamma2,alpha,
tolconjgrad,t_ext_hat,L,rowcol);
gamma_jm1 = 1; gamma_j = sqrt(z_j' * v_j); eta = gamma_j; s_jm1 =
0; s_j = 0; c_jm1 = 1;  c_j = 1; for j = 1:MaxIter
   z_j = z_j/gamma_j;
   Azj=Ax(z_j,nx,k_hat,alpha,B,D);
   d_j = z_j' * Azj';
   v_jp1 = Azj' - (d_j/gamma_j)*v_j - (gamma_j/gamma_jm1) * v_jm1;
   % use preconditioner
   [z_jp1] = precond(xprecond,v_jp1,D,B,gamma0,gamma1,gamma2,
   alpha,tolconjgrad,t_ext_hat,L,rowcol);
   gamma_jp1 = sqrt( z_jp1' * v_jp1 );
   a0 = c_j*d_j - c_jm1 * s_j * gamma_j;
   a1 = sqrt( a0^2 + gamma_jp1^2 );
   a2 = s_j * d_j + c_jm1 * c_j * gamma_j;
   a3 = s_jm1 * gamma_j;
   cjp1 = (a0/a1); s_jp1 = gamma_jp1/a1;
   w_jp1 = ( z_j - a3 * w_jm1 - a2 * w_j )/a1;
   u_j = u_jm1 + cjp1 * eta * w_jp1;
   eta =  - s_jp1 * eta;
   Au_j=Ax(u_j,nx,k_hat,alpha,B,D);
   res(j) = norm(b-Au_j');
   if res(j) < tol; iter=j; flag=0; break; end;
   % update for next iteration
   z_j = z_jp1;
   gamma_jm1 = gamma_j;  gamma_j = gamma_jp1;
   v_jm1 = v_j;  v_j = v_jp1;
   c_jm1 = c_j; c_j = cjp1;
   s_jm1 = s_j; s_j = s_jp1;
   w_jm1 = w_j; w_j = w_jp1;
   u_jm1 = u_j;
end
res=[norm(b),res];
% you can test the code by executing these lines
% clear
% n=1000;
% A=rand(n,n); A=A'*A; x=ones(n,1); b=A*x; M =diag(diag(A)); x0 = sparse(n,1);
% max = 2;   tol=1e-14;
% [u_j,res,iter,flag] = minres_OK(A,b,x0,max,tol,M);
% u_j;
% % plot(log(resP))
% figure
% plot(log(res));grid on;
% iter
% [xminres,flag,relres] = minres(A,b,tol,max,M);
% [xminres,u_j]
% [norm(xminres-u_j),norm(xminres-x),norm(x-u_j)]


%----------------------------precond----------------
function [y]
=precond(xprecond,x,D,B,gamma0,gamma1,gamma2,alpha,
tolconjgrad,t_ext_hat,L,rowcol);
m=size(B,1);  n=size(B,2); if xprecond == 0
    y = x;
else
  x1=x(1:m);
  x2=x(m+1:n+m);
  y1=D\x1;
  y1=y1/(alpha*gamma1);
  x2new = x2/gamma2;
  y2 = conjgrad(x2new,D,B,gamma0,gamma1,gamma2,alpha,tolconjgrad,
  t_ext_hat,L,rowcol);
  y=[y1;y2];
end
%---------------------------conjgrad--------------------------------
function x =
conjgrad(b,D,B,gamma0,gamma1,gamma2,alpha,tol,t_ext_hat,L,rowcol);

  n = 6000;
  m = 8000;
  A = randn(n,m);
  A = A * A';
  b = randn(n,1);
  tic, x = conjgrad(A,b); toc
  norm(A*x-b)

    if nargin<3
        tol=1e-10;
    end
    x = b;
    [Ax] = p2matrixvec(x,D,B,gamma0,gamma1,gamma2,alpha,
    t_ext_hat,L,rowcol);
    r = b - Ax;
    if norm(r) < tol
        return
    end
    y = -r;
    [Ay] = p2matrixvec(y,D,B,gamma0,gamma1,gamma2,alpha,
    t_ext_hat,L,rowcol);
    z = Ay;

    s = y'*z;

    t = (r'*y)/s;

    x = x + t*y;


    for k = 1:100;

       r = r - t*z;

       if( norm(r) < tol )

            return;

       end

       BB = (r'*z)/s;

       y = -r + BB*y;


       [Ay] = p2matrixvec(y,D,B,gamma0,gamma1,gamma2,alpha,

       t_ext_hat,L,rowcol);

       z = Ay;

       s = y'*z;

       t = (r'*y)/s;

       x = x + t*y;

    end

 end

%--------------------p2matrixvec--------------------------

function[w]=p2matrixvec(r,D,B,gamma0,gamma1,gamma2,alpha,

t_ext_hat,L,rowcol);

n = size(B,2); nx = sqrt(n);

% % -----------  PT preconditioner

 Lr = L*r;
```

```
Tr=restrict(ifft2(t_ext_hat.*fft2(extend(reshape(r,nx,nx))))); w =
Tr(:)+alpha*Lr ;
%%-------------PC and PS--Precond--just diff c---------
c=rowcol; Cr=ifft2(abs(c).^2.*fft2(reshape(r,nx,nx))); Cr=Cr(:);
w= Cr+alpha*Lr;
%------------------------------------------------------------------
%-------decoupled system stochastic Darcy------------------------
Ainv=inline('1','x','y'); global x_int w_int
load mesh1024       %------open this
% pdemesh(p,e,t)
 bo_triangles
M=6; d=4; Naxi=factorial(M+d)/(factorial(M)*factorial(d));
expect_psi = zeros(Naxi,1); expect_psi(1)=1;

[G0] = create_G0_uniform(p); [G1] = create_Gm_uniform(p,1);
[G2]=create_Gm_uniform(p,2); [G3] = create_Gm_uniform(p,3);
[G4]=create_Gm_uniform(p,4); [G5] = create_Gm_uniform(p,5);
[G6]=create_Gm_uniform(p,6);
% tau1=eig(inv(G0)*G1);
% tau2=eig(inv(G0)*G2);
% tau3=eig(inv(G0)*G3);
% tau4=eig(inv(G0)*G4);
A0 = StiffMat2D_m0(p,t); B = SecondMat2D(p,t);
f_vec=LoadVec2D(p,t); B5=B(5,:); mold=size(B,1);
B=B([1:4,6:mold],:); f_vec = f_vec([1:4,6:mold]); m=size(B,1);
At=[A0,B';B,sparse(m,m)]; b = [-B5';f_vec]; n=size(A0,1);
[At,b]=impose_boundary(vert,horz,At,b,n,m); A0 = At(1:n,1:n);
```

```
f_vec = b(n+1:n+m);
%------------------------------------------------------------------
x=[-0.93246951,-0.66120939,-0.23861919,0.23861919,0.66120939,0.93246951]';
x_int=(x+1)/2;
w_int=[0.17132449,0.36076157,0.46791393,0.46791393,0.36076157,0.17132449]';
x = x_int; w = w_int;

% we find the eigen pairs of the
%     integral operator

[tm,lm] = calculte_eig(x,w);
[lms,i] = sort(lm,'descend');

% --- we compute A_m for m=1:4
A1 = StiffMat2D(1,tm(:,i(1)),lms(1),p,t);

A2 =StiffMat2D(2,tm(:,i(2)),lms(2),p,t);

A3 = StiffMat2D(3,tm(:,i(3)),lms(3),p,t);

A4 = StiffMat2D(4,tm(:,i(4)),lms(4),p,t);

A5 = StiffMat2D(5,tm(:,i(5)),lms(5),p,t);

A6 = StiffMat2D(6,tm(:,i(6)),lms(6),p,t);

% ----Here we impose the bounday condition to A1,..,AM ----
```

```
At1=[A1,B';B,sparse(m,m)]; [At1]=

impose_boundary2(vert,horz,At1,n,m); A1 = At1(1:n,1:n);


At2=[A2,B';B,sparse(m,m)]; [At2]=

impose_boundary2(vert,horz,At2,n,m); A2 = At2(1:n,1:n);


At3=[A3,B';B,sparse(m,m)]; [At3]=

impose_boundary2(vert,horz,At3,n,m); A3 = At3(1:n,1:n);


At4=[A4,B';B,sparse(m,m)]; [At4]=

impose_boundary2(vert,horz,At4,n,m); A4 = At4(1:n,1:n);


At5=[A5,B';B,sparse(m,m)]; [At5]=

impose_boundary2(vert,horz,At5,n,m); A5 = At5(1:n,1:n);


At6=[A6,B';B,sparse(m,m)]; [At6]=

impose_boundary2(vert,horz,At6,n,m); A6 = At6(1:n,1:n);
%%-----------------------------------------------
B_hat = kron(G0,B);


A_hat = kron(G0,A0) + kron(G1,A1) + kron(G2,A2)+kron(G3,A3) +

kron(G4,A4)+kron(G5,A5)+kron(G6,A6);


f_vec_hat = kron(expect_psi',f_vec);


mm = size(B_hat,1);
```

```
nn = size(B_hat,2);

At=[A_hat,B_hat';B_hat,sparse(mm,mm)];

bb =sparse(nn+mm,1);

bb(nn+1:nn+mm) = f_vec_hat; D_hat = kron(G0,A0); S0=B*inv(A0)*B';
S0_hat = kron(G0,S0);
L = Grad_matrix(p,t,Ainv); mL=size(L,1);

L = L([1:4,6:mL],[1:4,6:mL]);

SL_hat = kron(G0,L); Gm = pres_massmat(p,t);

Gm(5,:)=[]; Gm(:,5)=[];

M = MassMat(p,t); M(5,:)=[]; M(:,5)=[];

SN_hat=kron(G0,Gm+M);
% SE_hat=B_hat*inv(A_hat)*B_hat';
P=[A_hat,sparse(nn,mm);sparse(mm,nn),SL_hat];
%%--------------------------------------------------------------
%[x_s,res1,iter1] = Pmyminres(At,bb,u0,maxit,tol,mm,nn,AZ,SZ);
%[x_s,res,iter,flag] = pminres(At,bb,u0,maxit,tol,M);
%%---------------------------------------------------------------
%P = [A0, zeros(n,m);zeros(m,n),Gm+M];
%P = [A0, zeros(n,m);zeros(m,n),L];
```

```
%%-----------------------------------------------------------
u0 = sparse(nn+mm,1); maxit=100; tol=1e-8;


tic
[xs,res1,iter1]=
Pmyminres(At,bb,u0,maxit,tol,mm,nn,A_hat,SL_hat); xs_v=xs(1:nn);
xs_p=xs(nn+1:nn+mm);
toc



for i=1:Naxi
        sta=(i-1)*m+1; %started point
        en=sta+m-1;  %end point
        pres(1:m,i)=xs_p(sta:en)';
end



for i=1:Naxi
    presnoze=pres(1:m,i); %pressure is the mean of the first column
    pres5=[presnoze(1:4,:);0;presnoze(5:m,:)];
    [intp]=integralp(t,p,pres5);
     uniqepres=pres5-intp;
     pres(1:m+1,i)=uniqepres;
end

meanp=pres(:,1);
for i=1:m+1
```

```
       varp(i)=var(pres(i,:));
end varp=varp';
%%------------------------------------------------------------
xxx=0:0.01:1; yyy=0:0.01:1;
meanpressure_matrix=tri2grid(p,t,meanp,xxx,yyy);


figure contourf(meanpressure_matrix,100);shading flat;colorbar;
figure mesh(xxx,yyy,meanpressure_matrix);
%%-------------------------------------------------------------
varpressure_matrix2=tri2grid(p,t,varp,xxx,yyy); figure
mesh(xxx,yyy,varpressure_matrix2);
figure
contourf(varpressure_matrix2,100);shading flat;colorbar;
%%----------------------------------------------------------
% [intp]=integralp(t,p,meanp); % to check is  the integral of p=0
vx = -xs_v(1:2:end);  %%%%%%%%%%vx are the odd components
vy = -xs_v(2:2:end);
n_nodes = size(p,2);
vx_node = zeros(n_nodes,Naxi); %it is  zero matrix
vy_node = zeros(n_nodes,Naxi);


for i = 1:n/2
for j = 1:3
        node_no = t(j,i);
        vx_node( node_no, : ) = vx_node( node_no, : ) + vx(i, :);
        vy_node( node_no, : ) = vy_node( node_no, : ) + vy(i, :);
end
```

```
end
vx_node=vx_node/6; %it is the mean
vy_node=vy_node/6; vx_node(5,:)=(vx_node(5,:)*6)/4;
vy_node(5,:)=(vy_node(5,:)*6)/4;
%all tringles have 6 nodes expet at the center (5) and the boundary(3)
vx_node(1,:)=(vx_node(1,:)*6)/2; vy_node(1,:)=(vy_node(1,:)*6)/2;
vx_node(2,:)=(vx_node(2,:)*6)/2; vy_node(2,:)=(vy_node(2,:)*6)/2;
vx_node(3,:)=(vx_node(3,:)*6)/2; vy_node(3,:)=(vy_node(3,:)*6)/2;
vx_node(4,:)=(vx_node(4,:)*6)/2; vy_node(4,:)=(vy_node(4,:)*6)/2;
bond_node = setdiff(e(1,:),[1,2,3,4]); nb = length(bond_node);
for
i=1:nb
    node_nm = e(1,i);
    vx_node(node_nm,:)=(vx_node(node_nm,:)*6)/3;
    vy_node(node_nm,:)=(vy_node(node_nm,:)*6)/3;
end
%%------------------------------------------------------------
xxx=0:0.01:1; yyy=0:0.01:1; meanvx_node=mean(vx_node,2);
meanvx_matrix=tri2grid(p,t,meanvx_node,xxx,yyy); figure
mesh(xxx,yyy,meanvx_matrix); figure
contourf(meanvx_matrix,100);shading flat;colorbar;
%%------velocities mean plot-------------------
  meanvy_node=mean(vy_node,2);
 meanvy_matrix=tri2grid(p,t,meanvy_node,xxx,yyy);
 mesh(xxx,yyy,meanvy_matrix);
 figure
contourf(meanvy_matrix,100);shading flat;colorbar;
```

```
%%------------------------------------------------------------
var_vx_node=sparse(n_nodes,1); var_vy_node=sparse(n_nodes,1);
for
i=1:n_nodes
    var_vx_node(i)=var(vx_node(i,:));
    var_vy_node(i)=var(vy_node(i,:));
end
%%----------velocitese var plot--------
varvx_matrix=tri2grid(p,t,var_vx_node,xxx,yyy);
contourf(varvx_matrix,100);shading flat;colorbar; figure
varvy_matrix=tri2grid(p,t,var_vy_node,xxx,yyy);
contourf(varvy_matrix,100);shading flat;colorbar;

figure quiver(meanvx_matrix,meanvy_matrix)
%%------------------------------------------------------
function [G0] = create_G0_uniform(p)
MM=6; p=4;
%here we use the complete polynomial space and uniform random field
vp=0:p;
[X1,X2,X3,X4,X5,X6] = ndgrid(vp,vp,vp,vp,vp,vp);
%[X1,X2,X3] = ndgrid(vp,vp,vp);
mul=[X1(:),X2(:),X3(:),X4(:),X5(:),X6(:)];
%mul=[X1(:),X2(:),X3(:)];
basis_deg=sum(mul,2)<p+1;
row=find(basis_deg==1);
order=mul(row,:); N_xi=length(row);
G0=zeros(N_xi,N_xi);
```

```
for i = 1: N_xi
    ri = order(i,:);
    value2 = 1;
for is=1:MM value2 = value2 * 1/(2*ri(is)+1); G0(i,i) = value2;
end end spy(G_0)
%%----------------------------------------------------------
function [Gm] = create_Gm_uniform(p,m) MM=6;
p=4;
%here we use the complete polynomial space and uniform random field
vp=0:p;
%[X1,X2] = ndgrid(vp,vp);
[X1,X2,X3,X4,X5,X6] = ndgrid(vp,vp,vp,vp,vp,vp);
%mul=[X1(:),X2(:)];
mul=[X1(:),X2(:),X3(:),X4(:),X5(:),X6(:)];
basis_deg=sum(mul,2)<p+1;
row=find(basis_deg==1); order=mul(row,:); N_xi=length(row);
Gm=zeros(N_xi,N_xi);
%m=1 gives G1 and m=2 gives G2 and so on
similar = setdiff([1:MM],m);
for i = 1: N_xi
    ri = order(i,:);
    for j = 1:N_xi
    ci = order(j,:);
    if  ri(1,similar) == ci(1,similar) & (ri(1,m)-ci(1,m)) == 1;
    value1=1;
for is=1:MM
```

```
    value1 = value1 * 1/(2*ri(is)+1);
    [i,j,value2,ri,ci];
end
    down = ri(m)/(2*ri(m)-1);
    value = value1*down;
    Gm(i,j) =value;
end


    if  ri(1,similar) == ci(1,similar) & (ri(1,m)-ci(1,m)) == -1;
    value2=1;
for is=1:MM
    value2 = value2 * 1/(2*ri(is)+1);
    [i,j,value2,ri,ci];
end
    down1 = ci(m)/((2*ci(m)-1)*(2*ci(m)+1));
    down2=(2*ri(m)+1);
    value2 = value2*down1*down2;
    Gm(i,j) =value2;
    [i,j,value2];
end
end
end
% spy(G_m)
% size(G_m)
% eig(G_m)
%%----------------------------------------------------------------
function A = StiffMat2D_m0(p,t)
```

```
Ainv=inline('1','x','y');

nt=size(t,2); n=2*nt; A=sparse(n,n);

for k=1:nt

    loc2glob=t(1:3,k);

    x=p(1,loc2glob);

    y=p(2,loc2glob);

    area=polyarea(x,y);

    Avec=[Ainv(x(1),y(1)),Ainv(x(2),y(2)),Ainv(x(3),y(3))];

    sumAvec=sum(Avec)*area/3;

    A(2*k-1,2*k-1)=sumAvec;

    A(2*k,2*k)=sumAvec;

end

%%----------------------------------------------------------------

function B = SecondMat2D(p,t) np = size(p,2); nt = size(t,2);

n=2*nt; m=np; B = sparse(m,n);


for k = 1:nt

    loc2glb = t(1:3,k); % local-to-global map

    x = p(1,loc2glb); % node x-coordinates

    y = p(2,loc2glb); % node y-

    [area,b,c] = Gradients(x,y);

    col1=2*k-1;

    col2=2*k;

    B(loc2glb,col1)=-b'*area;

    B(loc2glb,col2)=-c'*area;

end
```

```
%%----------------------------------------------------------
function F = LoadVec_5spot(p,t) f=inline('1','x','y');
np=size(p,2); nt = size(t,2); F = zeros(np,1);
F(1:4)=0.0043;  %%%%%%%%2/3 *area (0.0064)
F(5)=-6.8379e-004;  %%%%%%%%-4/3 *are (5.1284e-004)
%%------------------------------------------------------
function [At,b]= impose_boundary(vert,horz,At,b,n,m)
% -----------Here we impose the bounday condition--------
% no flow on the boundary i.e n . u = 0
nh = length(horz); nv = length(vert);

for i=1:nh
    irow = 2*horz(i);
    At(irow,:) = sparse(1,n+m);
    At(:,irow) = sparse(n+m,1);
    b(irow) = 0;
    At(irow,irow) = 1;
end

for i=1:nv
    irow = 2*(vert(i)-1)+1;
    At(irow,:) = sparse(1,n+m);
    At(:,irow) = sparse(n+m,1);
    b(irow) = 0;
    At(irow,irow) = 1;
end
```

```
%%--------------------------------------------------

function [tm,lm] = calculte_eig(x,w) n = length(x);


for i=1:n

for j=1:n

        ind = (i-1)*n + j;

        P(ind,1) = x(j);

        P(ind,2) = x(i);

        Wv(1,ind) = w(i)*w(j);

        % here since we have dauble integral and function w.r.t x and y


end

end


for k = 1:n^2

for l = 1:n^2

        Pk =[P(k,1),P(k,2)];

        Pl =[P(l,1),P(l,2)];

        C(k,l) = mycov(Pk,Pl);

end

end


W=[]; for i=1:n^2

    W = [W;Wv];

end


K = C.*W/4;  %4 came from the transition formula (0,1) into (-1,1)
```

```
[tm,lm1] = eig(K);


lm = diag(lm1);


%%----------------------------------------------------------
function [value] = mycov(x,y)
r = norm(x-y);
% -------- covariance in (2.9b) ---------
tao=1; sigma=0.1; value = sigma^2*exp(-r/tao) ;
%%------------------------------------------------
function Gm = pres_massmat(p,t) np = size(p,2); nt = size(t,2); A
= sparse(np,np); for K = 1:nt
    loc2glb = t(1:3,K); % local-to-global map
    x = p(1,loc2glb); % node x-coordinates
    y = p(2,loc2glb); % node y-
    [area,b,c] = Gradients(x,y);
    AK = (b*b'+c*c')*area; % element stiff mat
    A(loc2glb,loc2glb) = A(loc2glb,loc2glb)+ AK;
    % add element stiffnesses to A
end Gm=A;
%%--------------------------------------------------------------------
function M = MassMat(p,t) np = size(p,2); nt = size(t,2); alpha=4;
M = sparse(np,np); for K = 1:nt
    loc2glb = t(1:3,K); % local-to-global map
    x = p(1,loc2glb); % node x-coordinates
    y = p(2,loc2glb); % node y-
    [area,b,c] = Gradients(x,y);
    MK = [2 1 1;
              1 2 1;
              1 1 2]/12*area; % element mass matrix
    M(loc2glb,loc2glb) = M(loc2glb,loc2glb)+ MK;
end

%%-----------------------------------------------------------------
%clear
%load smallmesh
%load mesh_sample
%generate_mesh
%%%%%%%%%%%%%%we need to load M2p1 to work this program

Ainv=inline('1','x','y'); global x_int w_int

%load mesh_256
load mesh1024       %------open this
% load small_mesh
%load 5_Spot_Mesh
pdemesh(p,e,t) bo_triangles

% -----   change here 81 is the size of n_axi
G0=eye(81); Generate_G_matrices_M4 A0 = StiffMat2D_m0(p,t);

B = SecondMat2D(p,t);

f_vec = LoadVec2D(p,t);
```

```
% Modify the matrix B (has no full rank)
% so that the problem has a unique solution
% delete one row number 5 from B

B5=B(5,:); mold=size(B,1);
 B= B([1:4,6:mold],:);
 f_vec = f_vec([1:4,6:mold]);
 m=size(B,1);

 At=[A0,B';B,sparse(m,m)];
 b = [-B5';f_vec];

 n=size(A0,1);

% Modify the matrix B
% so that the problem has a unique solution
% delete four rows from B (1,2,3,4) corners ( production wells)

%  mold=size(B,1);
%  B1=B(1,:);B2=B(2,:);B3=B(3,:);B4=B(4,:);
%  B= B([5:mold],:);
%  f_vec = f_vec([5:mold]);
%  m=size(B,1);
%
%  At=[A0,B';B,sparse(m,m)];
%  b = [-B1'-B2'-B3'-B4';f_vec];
% ----------Here we impose the bounday condition--------

% no flow on the boundary i.e n . u = 0

[At,b]= impose_boundary(vert,horz,At,b,n,m);



A0 = At(1:n,1:n); f_vec = b(n+1:n+m);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x=[-0.93246951,-0.66120939,-0.23861919,0.23861919,0.66120939,0.93246951]';

x_int=(x+1)/2;

w_int=[0.17132449,0.36076157,0.46791393,0.46791393,0.36076157,0.17132449]';

x = x_int; w = w_int;


% ----------  we find the eigen pairs of the

%     integral operator

%

[tm,lm] = calculte_eig(x,w); [lms,i] = sort(lm,'descend');



% --- we compute A_m for m=1:4



% -----   change here --- find A3 A4

A1 = StiffMat2D(1,tm(:,i(1)),lms(1),p,t); A2 =

StiffMat2D(2,tm(:,i(2)),lms(2),p,t); A3 =

StiffMat2D(3,tm(:,i(3)),lms(3),p,t); A4 =
StiffMat2D(4,tm(:,i(4)),lms(4),p,t);

% -----   change here --- find A3 A4
% -----------Here we impose the bounday condition to A1,..,A4 --------



%we comment below to calculate the eigen_value theorm and for solution u
%shold to remove the comment

At1=[A1,B';B,sparse(m,m)]; [At1]=
impose_boundary2(vert,horz,At1,n,m); A1 = At1(1:n,1:n);
At2=[A2,B';B,sparse(m,m)]; [At2]=
impose_boundary2(vert,horz,At2,n,m); A2 = At2(1:n,1:n);

At3=[A3,B';B,sparse(m,m)]; [At3]=
impose_boundary2(vert,horz,At3,n,m); A3 = At3(1:n,1:n);
At4=[A4,B';B,sparse(m,m)]; [At4]=
impose_boundary2(vert,horz,At4,n,m); A4 = At4(1:n,1:n);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = Grad_matrix(p,t,Ainv); mL=size(L,1); L =
L([1:4,6:mL],[1:4,6:mL]); u0=zeros(n+m,1); maxit=100; tol=1e-5; Gm
= pres_massmat(p,t); Gm(5,:)=[]; Gm(:,5)=[]; M = MassMat(p,t);
M(5,:)=[]; M(:,5)=[];
%P = [A0, zeros(n,m);zeros(m,n),Gm+M];
%P = [A0, zeros(n,m);zeros(m,n),L];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pres=[]; vel=[]; [mean_psi]=psi_ex();

b1_vec = f_vec(1:m); res = zeros(81,50); iter_vec = zeros(81,1);
for i=1:81;
    AZ=A0+G(1,i)*A1+G(2,i)*A2+G(3,i)*A3+G(4,i)*A4;
    At=[AZ,B';B,sparse(m,m)];
    SZ=B*inv(AZ)*B';
    fZ= mean_psi(i) * b1_vec;
    bZ=[sparse(n,1);fZ];
%     xs = At\bZ;

mm=m; nn=n;
% [xs,res1,iter1] = Pmyminres(At,bZ,u0,maxit,tol,mm,nn,A0,L);
%[xs,res1,iter1] = Pmyminres(At,bZ,u0,maxit,tol,mm,nn,A0,Gm+M);
[xs,res1,iter1] = Pmyminres(At,bZ,u0,maxit,tol,mm,nn,AZ,SZ);
res(i,1:iter1) = res1; iter_vec(i) = iter1; [i,iter1]
    xs_v=xs(1:n);
    xs_p=xs(n+1:n+m);
    vel = [vel;xs_v];
    pres =[pres;xs_p];
end max_iteration = max( iter_vec ); for k=1:max_iteration
   big_res(k) = norm( res(:,k));
end plot(log(big_res/big_res(1)))
%-------------
mm=81*m; nn=81*n;
```

```
 ww = reshape(vel,n,81);
 pp = reshape(pres,m,81);
 pp = [pp(1:4,:);zeros(1,81);pp(5:m,:)];
 %%%%%%%%%we make the row number5=0 to make B has full rank%%%%%%%%%
 vx = ww(1:2:nn/81,:);  %%%%%%%%%%%vx are the odd components
 vy = ww(2:2:nn/81,:);   %%%%%%%%%%vy are the even components
[meanp] = mean_clc(pp,mean_psi); [intp]=integralp(t,p,meanp);
meanp=meanp-intp; [meanvx] = mean_clc(vx,mean_psi); [meanvy] =
mean_clc(vy,mean_psi);




n_nodes = size(p,2);
meanvx_node=zeros(n_nodes,1);  %%%mean of vx over the nods
meanvy_node=zeros(n_nodes,1);  %%%mean of vy over the nods

vx_node = zeros(n_nodes,81); vy_node = zeros(n_nodes,81); for i =
1:n/2
    for j = 1:3
        node_no = t(j,i);
        meanvx_node( node_no ) = meanvx_node( node_no ) + meanvx(i);
        meanvy_node( node_no ) = meanvy_node( node_no ) + meanvy(i);

        vx_node( node_no, : ) = vx_node( node_no, : ) + vx(i, :);
        vy_node( node_no, : ) = vy_node( node_no, : ) + vy(i, :);
    end
end meanvx_node=meanvx_node/6;
```

```
meanvx_node(5)=(meanvx_node(5)*6)/4;
%each nod has 6 triangles just the nod number 5 has 5 triangles
meanvy_node=meanvy_node/6; meanvy_node(5)=(meanvy_node(5)*6)/4;

vx_node=vx_node/6; vx_node(5,:)=(vx_node(5,:)*6)/4;
vy_node=vy_node/6; vy_node(5,:)=(vy_node(5,:)*6)/4;
%%%%%%%%%%%%%%%%%%%%%%%%%we plot the mean of the
%%%%%%%%%%%%%%%%%%%%%%%%%%pressure%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=0:0.01:1; y=0:0.01:1; pressure_matrix=tri2grid(p,t,meanp,x,y);
figure mesh(x,y,pressure_matrix); figure
contourf(pressure_matrix,100);shading flat;colorbar;
%%%%%%%%%%%%%%%%%%%%%%%%%plot the mean of vx and vy
xxx=0:0.01:1; yyy=0:0.01:1;
meanvx_matrix=-tri2grid(p,t,meanvx_node,xxx,yyy); figure
mesh(xxx,yyy,meanvx_matrix); figure
contourf(meanvx_matrix,100);shading flat;colorbar;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 figure
 meanvy_matrix=-tri2grid(p,t,meanvy_node,xxx,yyy);
mesh(xxx,yyy,meanvy_matrix); figure
contourf(meanvy_matrix,100);shading flat;colorbar;
%%%%%%%%%%%weplot the variance of pressure%%%%%%%%%%%%
[varp] = variance_calc(pp,meanp); [varvx] =
variance_calc(vx_node,meanvx_node); [varvy] =
variance_calc(vy_node,meanvy_node);
%----------------------------
x=0:0.01:1; y=0:0.01:1; pressure_matrix2=tri2grid(p,t,varp,x,y);
```

```
figure mesh(x,y,pressure_matrix2); figure
contourf(pressure_matrix2,100);shading flat;colorbar; figure
%%%%plot the variance of the velocity%%%%%%%%%%%%%%%%%%%%%%
xxx=0:0.01:1; yyy=0:0.01:1;
vx_matrix2=tri2grid(p,t,varvx,xxx,yyy); mesh(xxx,yyy,vx_matrix2);
figure contourf(vx_matrix2,100);shading flat;colorbar; figure
vy_matrix2=tri2grid(p,t,varvy,xxx,yyy); mesh(xxx,yyy,vy_matrix2);
figure contourf(vy_matrix2,100);shading flat;colorbar; figure
quiver(meanvx_matrix,meanvy_matrix)

%%------------------------------------------------------

function [intp]=integralp(t,p,mp); nelement=size(t,2); intp=0; for
k=1:nelement;
    loc2glb=t(1:3,k);
     x = p(1,loc2glb); % node x-coordinates
    y = p(2,loc2glb); % node y-
   area=polyarea(x,y);
    sumk=(sum(mp(loc2glb))*area)/3;
    intp=intp+sumk;
end

%%--------------------------------------------------------------------
function [mp] = mean_clc(pp,mean_psi);

ms = size(pp,1);
% EPSI(1)=sqrt(3)/(2*sqrt(2));
% EPSI(2)=1/sqrt(6);
% EPSI(3)=1/sqrt(6);
% index=0;
% for i= 1:3
%     for j=1:3
%         for k=1:3
%             for l=1:3
%                 index = index +1;
%                 EPSIY(index)=EPSI(i)*EPSI(j)*EPSI(k)*EPSI(l);
%             end
%         end
%     end
% end
% EPSIY
mp = zeros(ms,1); %mean pressure mp
Naxi=81; for i=1:Naxi
mp = mp + mean_psi(i)*pp(:,i); %meanpsi is the expection of apsis
end

%%-----------------------------------------------------
function [mean_psi]=psi_ex(); a1=sqrt(2/3);
 a2=1/sqrt(6);
  a3=1/sqrt(6);
  v1(1:27)=a1;
  v1(28:54)=a2;
  v1(55:81)=a3;
  z(1:9)=a1; z(10:18)=a2; z(19:27)=a3;
  v2=[z z z];
  w(1:3)=a1; w(4:6)=a2; w(7:9)=a3;
 wq=[w w w];
  v3=[wq wq wq];
  r=[a1 a2 a3];
  rr=[r r r];
  rrr=[rr rr rr];
  v4=[rrr rrr rrr];
  mean_psi=v1.*v2.*v3.*v4;
  %%%%%%%the expectition in the f_k on the right hand side %%%


%%-------------------------------------------------------
function [mean_psi]=psiexpection;
%a1=sqrt(3)/(2*sqrt(2));
 %a2=1/sqrt(6);
  %a3=1/sqrt(6);
  a1=2/3;
  a2=sqrt(5/18);
  a3=sqrt(5/18);
  v1(1:27)=a1;
  v1(28:54)=a2;
  v1(55:81)=a3;
  z(1:9)=a1; z(10:18)=a2; z(19:27)=a3;
  v2=[z z z];
  w(1:3)=a1; w(4:6)=a2; w(7:9)=a3;
 wq=[w w w];
```

```
  v3=[wq wq wq];

  r=[a1 a2 a3];

  rr=[r r r];

  rrr=[rr rr rr];

  v4=[rrr rrr rrr];

  mean_psi=v1.*v2.*v3.*v4;



%%--------------------------------------------------

function [value] = t_m(m,tm,lm,xx,yy); global x_int w_int

%sgma = 1;

sgma = 1; nq = length(x_int);


tm_matrix = reshape(tm,nq,nq); [X,Y] = meshgrid(x_int,x_int);

value1 = interp2(X,Y,tm_matrix,xx,yy,'spline');

%value = sgma*sqrt(lm)*value1;

value = sqrt(lm)*value1;


%%------------------------------------------------------------

function [varp] = variance_calc(pp,meanp);


ms = size(pp,1); %145

nxi = size(pp,2); varp = zeros(ms,1);


% ppsquare = pp.^2;

% term1 = sum(ppsquare,2);

% varp = term1 - meanp.^2;
```

```
for i=1:nxi diff(:,i)=pp(:,i)-meanp; sqdif=diff.^2; end

varp=mean(sqdif,2);

%%-----------------------------------------------
function [G_m] = create_Gm_Gaussian(p,m) M=2;
p=2;%here we use the complete polynomial space and Gaussian random field
vp=0:p;
[X1,X2] = ndgrid(vp,vp)
% if u change M=4 u must add [X1,X2,X3,X4] and ndgrid(vp,vp,vp,vp) and so on
%[X1,X2,X3,X4] = ndgrid(vp,vp,vp,vp)
mul=[X1(:),X2(:)];
%mul=[X1(:),X2(:),X3(:),X4(:)];
basis_deg=sum(mul,2)<p+1;
% wt we must do if we needd tensor polynomial space
row=find(basis_deg==1); order=mul(row,:) N_xi=length(row);
G_m=sparse(N_xi,N_xi);
m=1                             %m=1 gives G1 and m=2 gives G2 and so on
similar = setdiff([1:M],m); for i = 1: N_xi
    ri = order(i,:);
    for j = 1:N_xi
        ci = order(j,:);
        if  ri(1,similar) == ci(1,similar) & (ri(1,m)-ci(1,m)) == 1;
        for is=1:M
            value = value * factorial(  ri(is)  );
        end
        G_m(i,j) =value;
        end


        if  ri(1,similar) == ci(1,similar) & (ri(1,m)-ci(1,m)) == -1;
        value2=1;
        for is=1:M
            value2 = value2 * factorial(  ri(is)  );
            [i,j,value2,ri,ci]
        end
        down = factorial(ri(m));
        value2 = value2* factorial( ri(m)+1 )/down


            G_m(i,j) =value2
            [i,j,value2]
        end


    end
end spy(G_m)


%%---------------------------------------------
G(1,1:27)=0;
 G(1,28:54)=3/sqrt(5);
  G(1,55:81)=-3/sqrt(5);
V(1:9)=0;
 V(10:18)=3/sqrt(5);
  V(19:27)=-3/sqrt(5);
  G(2,1:81)=[V V
V];
 W(1:3)=0; W(4:6)=3/sqrt(5); W(7:9)=-3/sqrt(5); WW=[W W W];
G(3,1:81)=[WW WW WW];

Z=[0 3/sqrt(5) -3/sqrt(5)]; ZZ=[Z Z Z]; ZZZ=[ZZ ZZ ZZ];
G(4,1:81)=[ZZZ ZZZ ZZZ];
```

Bibliography

[1] Acar, R. and Vogel, C. R. (1994). Analysis of bounded variation penalty methods for ill-posed problems. *Inverse problems*, 10(6):1217.

[2] Achdou, Y., Bernardi, C., and Coquel, F. (2003). A priori and a posteriori analysis of finite volume discretizations of darcys equations. *Numerische Mathematik*, 96(1):17–42.

[3] Adams, R. A. and Fournier, J. J. (2003). *Sobolev spaces*, volume 140. Academic press.

[4] Agarwal, V. (2003). Total variation regularization and l-curve method for the selection of regularization parameter. *ECE599*, pages 1–31.

[5] Axelsson, O. and Neytcheva, M. (2006a). Eigenvalue estimates for preconditioned saddle point matrices. *Numerical Linear Algebra with Applications*, 13(4):339–360.

[6] Axelsson, O. and Neytcheva, M. (2006b). Eigenvalue estimates for preconditioned saddle point matrices. *Numerical Linear Algebra with Applications*, 13(4):339–360.

[7] Babuška, I., Nobile, F., and Tempone, R. (2007). A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, pages 1005–1034.

[8] Babuska, I., Tempone, R., and Zouraris, G. E. (2004). Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825.

[9] Babuška, I., Tempone, R., and Zouraris, G. E. (2005). Solving elliptic boundary value problems with uncertain coefficients by the finite element method: the stochastic formulation. *Computer methods in applied mechanics and engineering*, 194(12):1251–1294.

[10] Benzi, M. (2002). Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics*, 182(2):418–477.

[11] Benzi, M., Golub, G. H., and Liesen, J. (2005). Numerical solution of saddle point problems. *Acta numerica*, 14(1):1–137.

[12] Berisha, S. and Nagy, J. G. (2014). Iterative methods for image restoration. *Academic Press Library in Signal Processing: Image, Video Processing and Analysis, Hardware, Audio, Acoustic and Speech Processing*, 4:193–247.

[13] Bespalov, A., Powell, C. E., and Silvester, D. (2012). A priori error analysis of stochastic galerkin mixed approximations of elliptic pdes with random data. *SIAM Journal on Numerical Analysis*, 50(4):2039–2063.

[14] Biemond, J., Lagendijk, R. L., and Mersereau, R. M. (1990). Iterative methods for image deblurring. *Proceedings of the IEEE*, 78(5):856–883.

[15] Brenner, S. C. and Scott, R. (2008). *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media.

[16] Cao, Z.-H. (2002). A note on constraint preconditioning for nonsymmetric indefinite matrices. *SIAM Journal on Matrix Analysis and Applications*, 24(1):121–125.

[17] Cesari, L. (1937). *Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive*, volume 25. Nazionale Lincei R. Classe Sci. Fis. Mat. Nat.

[18] Chan, R. H. (1991). Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions. *IMA journal of numerical analysis*, 11(3):333–345.

[19] Chan, R. H., Chan, T. F., and Wong, C.-K. (1999a). Cosine transform based preconditioners for total variation deblurring. *Image Processing, IEEE Transactions on*, 8(10):1472–1478.

[20] Chan, R. H. and Ng, K.-P. (1993). Toeplitz preconditioners for hermitian Toeplitz systems. *Linear algebra and its applications*, 190:181–208.

[21] Chan, R. H.-F. and Jin, X.-Q. (2007). *An introduction to iterative Toeplitz solvers*, volume 5. SIAM.

[22] Chan, T. F. (1988). An optimal circulant preconditioner for toeplitz systems. *SIAM journal on scientific and statistical computing*, 9(4):766–771.

[23] Chan, T. F., Golub, G. H., and Mulet, P. (1999b). A nonlinear primal-dual method for total variation-based image restoration. *SIAM Journal on Scientific Computing*, 20(6):1964–1977.

[24] Chan, T. F. and Olkin, J. A. (1994). Circulant preconditioners for Toeplitz-block matrices. *Numerical Algorithms*, 6(1):89–101.

[25] de Sturler, E. and Liesen, J. (2005). Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems. part i: Theory. *SIAM Journal on Scientific Computing*, 26(5):1598–1619.

[26] Donatelli, M. (2005). A multigrid for image deblurring with tikhonov regularization. *Numerical linear algebra with applications*, 12(8):715–729.

[27] Donatelli, M. and Hanke, M. (2013). Fast nonstationary preconditioned iterative methods for ill-posed problems, with application to image deblurring. *Inverse Problems*, 29(9):095008.

[28] Eiermann, M., Ernst, O. G., and Ullmann, E. (2007). Computational aspects of the stochastic finite element method. *Computing and visualization in science*, 10(1):3–15.

[29] Elman, H., Furnival, D., and Powell, C. (2010a). H (div) preconditioning for a mixed finite element formulation of the diffusion problem with random data. *Mathematics of Computation*, 79(270):733–760.

[30] Elman, H., Furnival, D., and Powell, C. (2010b). H(div) preconditioning for a mixed finite element formulation of the diffusion problem with random data. *Mathematics of Computation*, 79(270):733–760.

[31] Elman, H., Silvester, D., and Wathen, A. (2014a). *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press.

[32] Elman, H. C., Silvester, D. J., and Wathen, A. J. (2014b). *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Oxford University Press.

[33] Ernst, O. G., Powell, C. E., Silvester, D. J., and Ullmann, E. (2009). Efficient solvers for a linear stochastic galerkin mixed formulation of diffusion problems with random data. *SIAM Journal on Scientific Computing*, 31(2):1424–1447.

[34] Ernst, O. G. and Ullmann, E. (2008). On stochastic galerkin matrices. In *preparation.* Citeseer.

[35] Ernst, O. G. and Ullmann, E. (2010). Stochastic galerkin matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1848–1872.

[36] Evans, D. J. (1968). The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices. *IMA Journal of Applied Mathematics*, 4(3):295–314.

[37] Ewing, R. E. and Shen, J. (1993). A multigrid algorithm for the cell-centered finite difference scheme. In *NASA Conference Publication*, pages 583–583. NASA.

[38] Fairag, F. A. and Wathen, A. J. (2012). A block preconditioning technique for the streamfunction-vorticity formulation of the navier-stokes equations. *Numerical Methods for Partial Differential Equations*, 28(3):888–898.

[39] Fortin, M. and Brezzi, F. (1991). *Mixed and hybrid finite element methods*. New York: Springer-Verlag.

[40] Frauenfelder, P., Schwab, C., and Todor, R. A. (2005). Finite elements for elliptic problems with stochastic coefficients. *Computer methods in applied mechanics and engineering*, 194(2):205–228.

[41] Ganis, B., Klie, H., Wheeler, M. F., Wildey, T., Yotov, I., and Zhang, D. (2008). Stochastic collocation and mixed finite elements for flow in porous media. *Computer methods in applied mechanics and engineering*, 197(43):3547–3559.

[42] Ghanem, R. (1999). Ingredients for a general purpose stochastic finite elements implementation. *Computer Methods in Applied Mechanics and Engineering*, 168(1):19–34.

[43] Ghanem, R. G. and Kruger, R. M. (1996). Numerical solution of spectral stochas-

tic finite element systems. *Computer Methods in Applied Mechanics and Engineering*, 129(3):289–303.

[44] Ghanem, R. G. and Spanos, P. D. (1991). Stochastic finite elements a spectral approach. *Springer*.

[45] Girault, V. and Raviart, P.-A. (1986). Finite element methods for navier-stokes equations: theory and algorithms, vol. 5 of springer series in computational mathematics.

[46] Gittelson, C. J. (2010). Stochastic galerkin discretization of the log-normal isotropic diffusion problem. *Mathematical Models and Methods in Applied Sciences*, 20(02):237–263.

[47] Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.

[48] Graham, I. G., Scheichl, R., and Ullmann, E. (2013). Mixed finite element analysis of lognormal diffusion and multilevel monte carlo methods. *arXiv preprint arXiv:1312.6047*.

[49] Greenbaum, A. (1997). *Iterative methods for solving linear systems*, volume 17. Siam.

[50] Grimmett, G. and Stirzaker, D. (1992). *Probability and random processes*, volume 2. Oxford Univ Press.

[51] Groetsch, C. W. and Groetsch, C. (1993). *Inverse problems in the mathematical sciences*, volume 52. Springer.

[52] Hackbusch, W. (2012). *Iterative solution of large sparse systems of equations*, volume 95. Springer Science & Business Media.

[53] Hanke, M. and Hansen, P. C. (1993). Regularization methods for large-scale problems. *Survey on Mathematics for Industry*, 3(4).

[54] Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the national Bureau of standard*, 49(6):409–436.

[55] Ipsen, I. C. (2001). A note on preconditioning nonsymmetric matrices. *SIAM Journal on Scientific Computing*, 23(3):1050–1051.

[56] Jin, C., Cai, X.-C., and Li, C. (2007). Parallel domain decomposition methods for stochastic elliptic equations. *SIAM Journal on Scientific Computing*, 29(5):2096–2114.

[57] Keese, A. (2003). A review of recent developments in the numerical solution of stochastic partial differential equations (stochastic finite elements). *Scientific Computing*, 6.

[58] Keese, A. (2004). *Numerical Solutions of Systems with Stochastic Uncertainties: A General Purpose Framework for Stochastic Finite Elements.* Mechanik-Zentrum, Techn. Univ.

[59] Klawonn, A. (1998). Block-triangular preconditioners for saddle point problems with a penalty term. *SIAM Journal on Scientific Computing*, 19(1):172–184.

[60] Kouri, D. P. (2010). *Optimization governed by stochastic partial differential equations.* PhD thesis, Rice University.

[61] Krzyzanowski, P. (2001). On block preconditioners for nonsymmetric saddle point problems. *SIAM Journal on Scientific Computing*, 23(1):157–169.

[62] Liesen, J. and Strakos, Z. (2012). *Krylov subspace methods: principles and analysis*. Oxford University Press.

[63] Lin, F.-R. (2001). Preconditioners for block Toeplitz systems based on circulant preconditioners. *Numerical Algorithms*, 26(4):365–379.

[64] Lin, F.-R. and Wang, C.-X. (2012). BTTB preconditioners for bttb systems. *Numerical Algorithms*, 60(1):153–167.

[65] Loeve, M. (1978). Probability theory, vol. ii. *Graduate texts in mathematics*, 46:0–387.

[66] Mathelin, L., Hussaini, M. Y., and Zang, T. A. (2005). Stochastic approaches to uncertainty quantification in cfd simulations. *Numerical Algorithms*, 38(1-3):209–236.

[67] Matthies, H. G. and Bucher, C. (1999). Finite elements for stochastic media problems. *Computer Methods in Applied Mechanics and Engineering*, 168(1):3–17.

[68] Matthies, H. G. and Keese, A. (2005). Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12):1295–1331.

[69] Meurant, G. (1999). *Computer solution of large linear systems*, volume 59. Elsevier Amsterdam.

[70] Money, J. H. (2006). Variational methods for image deblurring and discretized picard's method. *Ph.D.Thesis, UniversityofKentucky, Department of Mathematics.*

[71] Murphy, M. F., Golub, G. H., and Wathen, A. J. (2000a). A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972.

[72] Murphy, M. F., Golub, G. H., and Wathen, A. J. (2000b). A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972.

[73] Ng, M. K. and Pan, J. (2014). Weighted Toeplitz regularized least squares computation for image restoration. *SIAM Journal on Scientific Computing*, 36(1):B94–B121.

[74] Notay, Y. (2010). An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, 37(6):123–146.

[75] Øksendal, B. (2003). *Stochastic differential equations*. Springer.

[76] Olshanskii, M. A. and Tyrtyshnikov, E. E. (2014). *Iterative methods for linear systems: theory and applications*. Univercity of Houston, Texas.

[77] Paige, C. C. and Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629.

[78] Perugia, I. and Simoncini, V. (2000). Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations. *Numerical linear algebra with applications*, 7(7-8):585–616.

[79] Pestana, J. and Wathen, A. J. (2015). Natural preconditioning and iterative methods for saddle point systems. *SIAM Review*, 57(1):71–91.

[80] Powell, C. E. and Ullmann, E. (2010). Preconditioning stochastic galerkin saddle point systems. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2813–2840.

[81] Riley, K. L. (1999). *Two-Level Preconditioners For Regularized Ill-Posed Problems*. PhD thesis, Montana State University-Bozeman.

[82] Roman, L. J. and Sarkis, M. (2006). Stochastic galerkin method for elliptic spdes: A white noise approach. *Discrete and Continuous Dynamical Systems-Series B*, 6(4):941.

[83] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

[84] Rusten, T. and Winther, R. (1992). A preconditioned iterative method for saddlepoint problems. *SIAM Journal on Matrix Analysis and Applications*, 13(3):887–904.

[85] Saad, Y. (2003). *Iterative methods for sparse linear systems.* Univercity of Minnesota, SIAM.

[86] Saad, Y. and Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869.

[87] Schwab, C. and Todor, R. A. (2006). Karhunen–loève approximation of random fields by generalized fast multipole methods. *Journal of Computational Physics*, 217(1):100–122.

[88] Serra, S. (1994). Preconditioning strategies for asymptotically ill-conditioned block Toeplitz systems. *BIT Numerical Mathematics*, 34(4):579–594.

[89] Silvester, D. and Wathen, A. (1994a). Fast iterative solution of stabilised stokes systems part ii: using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367.

[90] Silvester, D. and Wathen, A. (1994b). Fast iterative solution of stabilised stokes

systems part ii: using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367.

[91] Strang, G. (1986). A proposal for Toeplitz matrix calculations. *Studies in Applied Mathematics*, 74(2):171–176.

[92] Tikhonov, A. N. (1963). Regularization of incorrectly posed problems. In *Soviet Math. Dokl*, volume 4, pages 1624–1627.

[93] Traverso, L., Phillips, T., and Yang, Y. (2014). Efficient stochastic fem for flow in heterogeneous porous media. part 1: random gaussian conductivity coefficients. *International Journal for Numerical Methods in Fluids*, 74(5):359–385.

[94] Treves, F. (1967). Topological vector spaces, distributions and kernels. *Academic, New York*, (25).

[95] Turing, A. M. (1948). Rounding-off errors in matrix processes. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1(1):287–308.

[96] Tykhonov, A. (1963). Regularization of incorrectly posed problems. In *Soviet Math. Doklady*, volume 4, pages 1624–1627.

[97] Van der Vorst, H. A. (2003). *Iterative Krylov methods for large linear systems*, volume 13. Cambridge University Press.

[98] Vogel, C. R. and Oman, M. E. (1998a). Fast, robust total variation-based reconstruction of noisy, blurred images. *Image Processing, IEEE Transactions on*, 7(6):813–824.

[99] Vogel, C. R. and Oman, M. E. (1998b). Fast, robust total variation-based reconstruction of noisy, blurred images. *Image Processing, IEEE Transactions on*, 7(6):813–824.

[100] Wathen, A. (2015). Preconditioning. *Acta Numerica*, 24:329–376.

[101] Xiu, D. and Karniadakis, G. E. (2002). Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos. *Computer methods in applied mechanics and engineering*, 191(43):4927–4948.

[102] You, Y.-L. and Kaveh, M. (1996). Anisotropic blind image restoration. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 461–464. IEEE.

VITAE

- Adel Mohammed Yahya Al-Mahdi.
- Born in Ibb, Yemen on January 1, 1977.
- Received Bachelor of Science (BSc) degree in Mathematics (batch 2001-2002) from Ibb University, Ibb, Yemen in 2002.
- Appointed as a graduate assistant at Ibb University, college of Al-Nadera in 2003, and I am still working there as a faculty member.
- Received a scholarship from the Ministry of Higher Education and Ibb University to study MS degree at King Fahd University of Petroleum and Minerals, in 2007.
- Received Master of Since (MSc) degree in Mathematics from King Fahd University of Petroleum and Minerals, Dhahran, KSA, in 2011.
- Submitted this dissertation to fulfil the requirements of his PhD degree in Mathematics from King Fahd University of Petroleum and Minerals.
- **Publications**
  1. F. Fairag and A. Al-Mahdi. Performance of Three Preconditioners for Image Deblurring Problem in Primal-Dual Formulation. (Published) in Engineers and Computer Scientists 2015 Vol I.
  2. Ke, Chen, Faisal Fairag and Adel Al-Mahdi. Preconditioning Techniques for an Image Deblurring Problem. (Accepted) in Journal of Numerical Linear Algebra with Applications.
  3. F. Fairag and A. Al-Mahdi . Accelerating Image Deblurring Using Circulant Approximations. (Accepted) in IMECS 2015 edited book published by Springer.

- **Present Address:** Department of Mathematics and Statistics, King Fahd University of Petroleum and Minerals, P.O. Box 8585, Dhahran 31261, Saudi Arabia.
- **E-mail Address:** g200704510@kfupm.edu.sa
- **Permanent Address:** Department of Mathematics, Ibb University, Ibb, Yemen.
- **E-mail Address:** almahdi77@yahoo.com.